[redacted]

UNITED STATES DISTRICT COURT
SOUTHERN DISTRICT OF NEW YORK

In Re: OpenAI, Inc. Copyright Infringement Litigation This Document Relates To: All Cases	Case No. 1:25-md-3143 (SHS) (OTW

DECLARATION OF JOHN D. LAFFERTY, PH.D. IN SUPPORT OF MICROSOFT'S MOTIONS FOR SUMMARY JUDGMENT

[redacted]

I, John D. Lafferty, Ph.D., hereby declare as follows:

I. ASSIGNMENT

1. I am over the age of 18 and am competent to make this declaration.

2. I have been engaged by counsel for Microsoft to provide technical and statistical expert analysis in this matter, including to: (1) provide an overview of the technical concepts relating to Artificial Intelligence, Large Language Models ("LLMs"), retrieval-augmented generation ("RAG" or "grounding"), and the associated Microsoft product functionality, including model training, grounding, and output creation (Section IV); (2) analyze and provide an overview of the Microsoft Copilot Chat Log Data ("Chat Log Data"), which capture, among other things, the prompts, outputs, and grounding data for the version of the consumer Bing Chat/Microsoft Copilot product released February 2023 and made generally available from December 2023 through September 2024 ("Copilot") (Section V); (3) select a statistically robust sample from the Chat Log Data of a size sufficient for an assessment of the general usage patterns of Copilot (Sections VI–VIII); and (4) respond to certain opinions and analyses offered by Plaintiffs' experts concerning LLM training and memorization, grounding and retrieval, the claimed unique importance of news and book content, the Deucalion fine-tuning data, and the rates at which Copilot outputs contain Plaintiffs' content, including as presented in The Times and Daily News Plaintiffs' expert Dr. Tom Goldstein's supplemental analysis (Sections IX–XVI).

3. I submitted an opening expert report in this case on April 22, 2026, a rebuttal expert report on May 29, 2026, a reply expert report on June 26, 2026, a sur-reply expert report on July 28, 2026, and an errata to my sur-reply report on July 30, 2026. The opinions below are a subset of the opinions set forth in my expert reports. The statements in those reports and in this declaration are true and accurate to the best of my knowledge, information, and belief. Except as otherwise stated herein, I make this declaration based on my personal knowledge

[redacted]

and professional expertise, materials reviewed in this case (as cited herein), and research and analysis conducted by me or under my direction and supervision.

4. To support the development of my analyses, staff working under my direction provided research support and executed certain technical analyses under my supervision; I reviewed and adopted the analyses and results presented as my own. All opinions in this declaration are mine and mine alone. A copy of my current curriculum vitae is attached as Appendix A.

II. QUALIFICATIONS

5. I am the John C. Malone Professor in the Department of Statistics and Data Science at Yale, with a secondary appointment in Computer Science. I am also an Associate Director of the Wu Tsai Institute at Yale, a University-wide institute with the mission to understand human cognition and explore human potential by sparking interdisciplinary inquiry. I am also the Director of the Center for Neurocomputation and Machine Intelligence at the Wu Tsai Institute, which is focused on developing new computational frameworks for studying the brain. In addition, I am Co-Chair of the Steering Committee of the Yale Data Science Fellows Program, and a member of the Institute for the Foundations of Data Science at Yale.

6. I received a PhD in Mathematics from Princeton University in 1986, an MA in Mathematics from Princeton in 1984, and a BA in Mathematics from Middlebury College in 1982. In 1981 I studied mathematics as a Special Student at the Massachusetts Institute of Technology, and I received a national fellowship as a Beinecke Scholar that I held at MIT and Princeton.

7. Before joining Yale, I was the Louis Block Professor in the Department of Statistics and the Department of Computer Science at the University of Chicago, an Adjunct Professor at the Toyota Technological Institute of Chicago, and a faculty member at Carnegie Mellon University. At Carnegie Mellon, I was a faculty member in the Computer Science Department, the Department of

[redacted]

Statistics, and the Machine Learning Department, where I played a role in founding and directing the Machine Learning PhD program.

8. Before joining Carnegie Mellon, I was a Research Staff Member at the IBM Thomas J. Watson Research Center, where my work focused on statistical approaches to natural language processing, language modeling, machine translation, and speech recognition. Before this, I was an Assistant Professor and Benjamin Pierce Lecturer on Mathematics at Harvard University.

9. My core research focuses on machine learning, natural language processing, language models, high-dimensional data analysis, and the statistical foundations that underpin these topics, which have applications in many disciplines. Per Google Scholar, I have over 85,000 citations, an h-index of 83, which means 83 of my publications have been cited over 83 times, and an i10 index of 161, meaning that 161 of my publications have been cited more than 10 times. I have received numerous awards for my research in machine learning. In 2007 I was elected as a Fellow of the Institute for Electrical and Electronics Engineers (IEEE) for my contributions to statistical pattern recognition and statistical language processing. I have also received several awards recognizing seminal contributions to the field of machine learning by some of the most prestigious conferences in the field; these awards include the following: a) 2016 Test of Time Award from the International Conference on Machine Learning for my paper "Dynamic topic models" with David Blei. b) 2014 Test of Time Award from SIGIR for my paper "Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval," with William Cohen and Chengxiang Zhai. c) 2013 Test of Time Award from the International Conference on Machine Learning for my paper "Semi-supervised learning using Gaussian fields and harmonic functions," with Xiaojin Zhu and Zoubin Ghahramani. d) 2012 Test of Time Award from the International Conference on Machine Learning for my paper "Diffusion kernels on graphs and other discrete structures" with Risi Kondor. e) 2011 Test of Time Award from the International Conference on Machine Learning for my paper "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," with Andrew McCallum and Fernando Pereira.

[redacted]

10. Between 2000 and 2003 I was Co-PI (with Bruce Croft) of the project "A Unified Framework for Information Management Using Statistical Language Models," funded by ARDA, a US intelligence community agency, which developed some of the early foundations of using generative language modeling for search and retrieval of information. I was invited to give a tutorial on Probabilistic Language Modeling at NeurIPS in 1999.

11. I have served in several leadership roles in the machine learning research community. As examples, I served as Program Chair and General Chair of the Neural Information Processing Systems (NeurIPS) Conference in 2009 and 2010. I have also served on the Advisory Board and the Executive Board of the NeurIPS Foundation. As a Professor at Yale University, I teach classes such as Introductory Machine Learning and Intermediate Machine Learning. In 2024, my teaching at Yale was recognized by the Lee Dixon '63 Prize for Teaching Excellence.

12. I currently serve on the Executive Committee of the Institute for Social and Policy Studies (ISPS) at Yale, and as a member of the University AI Steering Committee. A current copy of my curriculum vitae is provided as Appendix A to this declaration.

13. As part of my research, I have trained language models, including models using both recurrent neural networks and the transformer architecture underlying modern LLMs with on the order of up to 3 billion parameters. In earlier work, including my work on language processing and information retrieval at IBM and Carnegie Mellon University, I trained n-gram language models and probabilistic grammars on large corpora. My opinions in this declaration are informed by this first-hand experience designing, training, evaluating, and using language models.

III. SUMMARY OF OPINIONS

14. This declaration proceeds as follows. Sections IV through VIII present my affirmative technical analysis: the scientific and technical background on Artificial Intelligence, LLMs, and grounding (Section IV); an overview of Copilot

[redacted]

and the Chat Log Data (Section V); the methodology for selecting a statistically robust sample of the Chat Log Data (Section VI); robustness checks on the sample size (Section VII); and the sample selection process and redaction of personally identifiable information (Section VIII). Sections IX through XVI respond to opinions of Plaintiffs' experts bearing on the same subjects: the objective of LLM development (Section IX); grounding and retrieval (Section X); the claimed unique functional importance of news and book content (Section XI); the Deucalion fine-tuning data (Section XII); the rates at which Copilot outputs contain text matching Plaintiffs' asserted works, and the flaws in Plaintiffs' experts' contrary analyses (Section XIII); the prevalence of news domains among the URLs retrieved by Copilot's grounding system (Section XIV); shared text between grounding data and outputs (Section XV); Plaintiffs' experts' theoretical claims that generalization requires memorization and that LLM training is equivalent to compression (Section XVI). Section XVII states my conclusions.

15. In Section IV.A., I provide an overview of the core concepts underlying Artificial Intelligence and Large Language Models, which have been built from decades of innovation in the fields of Computer Science and Natural Language Processing ("NLP"). These concepts provide the necessary scientific and technical background to understand the basic nature of large language models, creation of such models through a process commonly referred to with the over-arching term training, and the implementation of such models in a system or product like Copilot.

16. In Section IV.B., I describe the process of pre-training LLMs. During this process an extremely large volume and a wide variety of textual data is assembled and processed into a training "corpus," then used to train a neural network such that it has acquired a broad underlying set of statistical associations from that training corpus. The acquisition of these statistical associations is also referred to in the literature as "learning," although learning in the technical context is quite different from how humans learn. These statistical associations, acquired from processing a wide variety of texts and subject matters, result in factual information, aspects of semantics and syntax, and other properties of language

[redacted]

being exhibited by the models as they generate new text. The resulting pre-trained large language model has a wholly different purpose than any of the individual original texts. While a particular text may be read for entertainment or for information, a pre-trained LLM is a type of general purpose technology that can be used for a wide variety of creative, scientific, or information-centered tasks across information processing and retrieval, question answering, content generation, summarization, writing, coding, and translation, among others. This functionality is based upon the statistical associations and patterns of co-occurrence that appear across the full distribution of the training data, rather than being based in any single document or small collection of texts. I further describe the additional processes used by model developers to take a pre-trained model and to further develop specific capabilities, domains, or modes of reasoning. Again, this process uses text in a way that is designed to instill functional capabilities in an LLM, and is not dependent upon the content of any particular text.

17. In Section IV.C., I describe the additional processes of fine-tuning and post-training LLMs. These late-stage development processes are used to align a pre-trained model with human preferences, in order to follow specific instructions and exhibit desired characteristics, such as helpfulness and safety. I describe how each particular stage of development addresses LLM output generation and uses data that is meaningfully different from the data used during the pre-training process. I then summarize my opinions about the entire training process, concluding that properly trained LLMs are designed to generate outputs that are generally novel as (i) the objective of model training is to generalize so that the model can address new, previously unseen contexts, rather than to memorize specific instances of training data, (ii) the outputs are not fixed, but are created by sampling from probability distributions that are dynamically constructed, and learned over the entire distribution of data, and (iii) the textual material used for training is not stored by the model and reproduced during the ordinary operation of the model.

[redacted]

18. In Section IV.D., I describe how LLMs can use external data during generation in order to produce outputs. The process of using an LLM after it has been trained is commonly referred to as "inference." By using an external source of textual data as part of the prompt and output process during inferencing, an LLM-based product can use that external data to provide summaries and answers to particular questions. This use of external data at inferencing time is referred to as Retrieval-Augmented Generation ("RAG"), which is also referred to as "grounding" the output of the model in external data. Grounding in an LLM-based information retrieval system provides two clear advantages. First, it enables the product to refer to an external context for "ground truth," thus improving factual accuracy of the generated output. Second, grounding enables the model to refer to factual information that is outside of the datasets used in the training corpus. In particular, this enables the model to generate output based on facts and events occurring after the training process was completed, again enhancing the factual accuracy of the output generated by the model.

19. In Section V, I provide a brief overview of Copilot and Chat Log Data, including how it is stored, structured, and the fields contained within the dataset. It is my understanding that the Microsoft witnesses will provide more detailed background regarding the development and operation of that product. My overview description of Copilot is provided simply as background for describing my study and analysis of the output data generated by users of that product.

20. I offer my opinion in Section VI, VII, and VIII regarding the desired sampling approach for assessing the real-world uses of the Copilot product by consumers. In order to analyze the general usage patterns of Microsoft Copilot, I develop a statistically robust approach to establish an appropriate sample size and a sampling methodology to draw from the population of the Chat Log Data. I determine that an appropriate sample size for analyzing usage patterns is 30,000 conversations, drawn as a simple random sample using the conversation ID for selection.

[redacted]

21. My principal responsive opinions to Plaintiffs' Experts' opinions are as follows. First, LLM training does not store Plaintiffs' works as retrievable copies; model weights are distributed statistical parameters, and the objective of training is generalization, not reproduction. Second, grounding and retrieval do not establish that Copilot's user-facing output reproduced substantial portions of any Plaintiff's work; impressions, citations, and grounding-field entries measure retrieval, not output copying. Third, news and book content are not uniquely necessary to LLM capability; the attributes Plaintiffs' Experts identify are available across many sources and are not a function of copyright status. Fourth, the small number of Plaintiff-domain references in the Deucalion fine-tuning data does not establish that Plaintiff content caused Deucalion's improved performance. Fifth, Copilot's measured rate of output reproduction of Plaintiffs' content is very low even in Plaintiffs' Experts' own analyses, and lower still once their methodological choices are corrected and measured against a denominator representing the population of Copilot conversations. I describe each in more detail below and in the sections that follow.

22. In Section IX, I discuss how Plaintiffs' Experts mischaracterize the objective of LLM pretraining when describing next-token prediction, memorization, and inference. Next-token prediction is the mechanism used to update model parameters during training. It is not an instruction to build a system that stores and later reproduces training text. The objective of model development is generalization to new prompts and contexts, as reflected by held-out evaluation. Model weights are numerical parameters distributed across a machine learning network, which reflect statistical relationships learned during training rather than a table, index, file system, or database of training documents. Memorization can occur in limited circumstances, particularly where text is repeated, distinctive, or elicited through extraction-directed prompting, but Plaintiffs' Experts overstate its role in ordinary LLM operation.

23. In Section X, I discuss how Dr. Goldstein misrepresents grounding and retrieval as a mechanism for copying and conflates two distinct steps, grounding and generation, with the premise that each impression from Copilot reflects copying a

[redacted]

substantial portion of the webpage, including article content. Grounding (and consequently, an impression) means that information from a source may be retrieved, referenced, or provided as context to help formulate a response. It does not by itself establish that the user-facing output contains copied, reproduced, or regurgitated source material. Impressions, citations, context fields, and snippets may support that some content has been retrieved or processed in forming a response, but they do not establish that each impression involved copying any portion, let alone substantial portions, of a Plaintiff article.

24. In Section XI, I discuss how Plaintiffs' Experts overstate the unique functional importance of news and book-like content. News and books may have useful characteristics and attributes, including coherence, factual density, recency, editorial quality, domain coverage, long-form discourse, and linguistic diversity. Plaintiffs' Experts, however, conflate the idea that high-quality data can improve model performance with the different claim that news or books are uniquely necessary or materially responsible for LLM capability. The attributes Plaintiffs identify are not unique to news, books, copyrighted works, or any particular publisher's articles or books. Additionally, copyright status is a legal attribute, not a property of training data, and the ability to generate news-like or book-like text is consistent with learning register-, genre-, and discourse-level patterns across many examples.

25. In Section XII, I discuss the point that the small number of Plaintiff-domain references identified in the synthetic data Microsoft provided for fine-tuning Deucalion does not establish that Plaintiffs' works caused Deucalion's improved performance. The reported 45 nytimes.com references, including 5 Wirecutter references, must be evaluated against the full set of Deucalion fine-tuning data. Without an appropriate denominator, those counts do not establish that Plaintiff content made a measurable contribution to Deucalion's performance. In the same manner the portion of the Deucalion fine-tuning data generated by Microsoft was made available to Plaintiffs' Experts, this data was made available to my team who ran an analysis to calculate the corresponding denominator of messages, which was found to be 228,679, which demonstrates that the number of

[redacted]

messages that include a Plaintiff URL is only 0.02% of the Deucalion synthetic fine-tuning data.[1] More information-rich responses are consistent with post-training and fine-tuning designed to improve grounded answer quality, not with memorization or storage of Plaintiffs' works in model parameters.

26. In Section XIII, I describe how the Opposing Experts' Copilot regurgitation analyses show vanishingly low rates of regurgitation, even before correcting for the methodological issues in their reports. Dr. Goldstein's analysis specifically included flawed assumptions, such as 4-gram matching, allowing for double counting of matches, and grouping multiple articles together to count as a match, all of which incorrectly inflate the results. This is further evidenced by the fact that CIR's expert Dr. Emily Wenger's analysis and Class Plaintiffs' expert Dr. Shawn Shan's results agree with each other and are 450x – 15,000x smaller than Dr. Goldstein's results. However, even with these flawed assumptions, the reported rates are extremely low. Dr. Goldstein finds 4.5% regurgitation for News Plaintiffs' articles, Dr. Wenger finds 0.0096% regurgitation for Mother Jones, Reveal News, and The Intercept Asserted Works, and Dr. Shan finds 0.0003% regurgitation for the Books Plaintiffs' works.

27. Critically, the Plaintiffs' Experts analyses rely on an 8.2 million subset of Copilot conversations which contain the Plaintiff keyword or domains from their Asserted Works, rather than a random sample from the entire population of more than 1 billion Copilot conversations. That subset is therefore a biased sub-population for regurgitation-related estimates. Without even correcting for the methodological choices in Plaintiffs' Experts analyses, and rather just using the denominator of 1 billion conversations representing the population, as opposed to the biased denominator of the Copilot conversations including Plaintiff keyword or domains, the percentages drop to 0.9% for Dr. Goldstein's analysis, 0.00008% for Dr. Wenger's analysis, and 0.000002% for Dr. Shan's analysis.

[1] 45 / 228,679 = 0.0001967824 * 100 = 0.02%.

[redacted]

28. Further, I then correct for the methodological errors in the Plaintiffs' reports in my rebuttal analysis. I primarily focus on the Goldstein report as it contains the most egregious regurgitation rates and methodological errors. As a result, I show that the regurgitation rates are even lower than Dr. Goldstein's news regurgitation analysis at 0.02% for the 8.2 million subset of Copilot conversations which contain the Plaintiff keyword or domains and 0.004% for the 1 billion conversations, and 0.000003% for the Books datasets in the 8.2 million subset of and 0.0000004% for the 1 billion conversations. A summary of these analyses and results is presented below in Table 1 (Rebuttal).

Table 1 (Rebuttal): Lafferty and Plaintiffs' Expert Regurgitation Analyses Results

Expert (Data Field)	Match Percent (8.2M dataset)[2]	Extrapolated Match Percent (1B conversations)
News		
Goldstein[3] ([redacted])	4.5%[4]	0.9%[5]
Lafferty ([redacted])	**0.02%**	**0.004%**
Wenger ([redacted] and [redacted])[6,7]	0.0096%[8]	0.00008%[9]

[2] The Shan and Wenger analyses count conversations, while the Lafferty and Goldstein analyses count messages.

[3] *See* Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398.

[4] 8,583,560 / 191,442,357 = 0.0448. 0.0448 * 100 = 4.5%. (Where 191,442,357 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations, which contain the Plaintiff keyword and domains).

[5] 8,583,560 / 1,000,000,000 = 0.00858356. 0.00858356 * 100 = 0.9%. (Assuming 1B [redacted] messages).

[6] Wenger approximates the Copilot Chat Logs to contain 8.4 million conversations, which is roughly 200,000 more conversations than were produced. I apply a denominator of 8.2 million in this table for consistency. Dr. Wenger described that she received a set of output logs from Microsoft Copilot including full logs of approximately 8.4 million conversations between users and chatbots, as well as internal log messages saved by Copilot as it processed user queries. *See* Hurst Decl. Ex. 246, Wenger Opening Report p. 104. ("I also received a set of output logs from Microsoft Copilot. The data include full logs of approximately 8.4 million conversations between users and chatbots, as well as internal log messages saved by Copilot as it processed user queries.").

[7] Wenger analysis code in msft_postprocess.py at line 129 combined both [redacted] and [redacted] fields for n-gram analysis.

[8] Dr. Wenger's opening analysis estimates 280, 494, and 15 conversations present in the 8.2M Copilot conversations which contain the Plaintiff keywords or domains. 280 + 494 + 15 = 789. 789 / 8,195,308 = 0.000096. 0.000096 * 100 = 0.0096%.

[9] 789 / 1,000,000,000 = 0.000000789. 0.000000789 * 100 = 0.00008%.

[redacted]

Goldstein ([redacted])	11.4%[10]	1.1%[11]
Lafferty ([redacted])	**0.09%**	**0.009%**
Books		
Shan[12] ([redacted] and [redacted])	0.0003%[13]	0.000002%[14]
Lafferty ([redacted])	**0.000003%[15]**	**0.0000004%[16]**
Lafferty ([redacted])	**0.000008%[17]**	**0.0000008%[18]**

29. I find two fundamental problems with Dr. Goldstein's grounding and rebuttal analyses. First, Dr. Goldstein's grounding analysis using the Bell News List[19] and Pink Slime List[20] reports statistics that are artificially and inappropriately inflated. The analysis reports the share of conversations containing at least one domain-list hit, but that conversation-level denominator does not measure the prevalence of those sources among the URLs retrieved by Copilot's grounding

[10] 10,922,091 / 95,769,219 = 0.114. 0.114 * 100 = 11.4% (Where 95,769,219 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[11] 10,922,091 / 1,000,000,000 = 0.010922091. 0.010922091 * 100 = 1.1%. (Assuming 1B [redacted] messages).

[12] Hurst Decl. Ex. 241, Shan Opening Report, ¶156.

[13] 24 / 8,195,308 = 0.0000029285. 0.0000029285 * 100 = 0.0003%. Dr. Shan, applying his methodology to Microsoft's output logs, identified reproductions of 30 or more near-verbatim words from 10 Plaintiff works across 24 conversations and 24 chat responses. *See* Hurst Decl. Ex. 241, Shan Opening Report ¶156 ("Applying the methodology outlined in § IV supra, to Microsoft's output logs, I identified reproductions of 30 or more near-verbatim words as defined in Paragraph 40 from 10 Plaintiff works across 24 conversations and 24 chat responses").

[14] 24 / 1,000,000,000 = 0.000000024 * 100 = 0.000002%.

[15] 4 / 158,047,588 = 0.0000000253. 0.0000000253 * 100 = 0.000003% (Where 158,047,588 is the number of non-empty "bot" [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[16] 4 / 1,000,000,000 = 0.000000004. 0.000000004 * 100 = 0.0000004% (Assuming 1B [redacted] messages).

[17] 8 / 95,719,689 = 0.0000000835. 0.0000000835 * 100 = 0.000008% (Where 95,719,689 is the number of non-empty "bot" [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[18] 8 / 1,000,000,000 = 0.000000008. 0.000000008 * 100 = 0.0000008% (Assuming 1B [redacted] messages).

[19] The Bell News List is a list of news domains compiled from Palewire's homepages.news, the Medill State of Local News Project 2025, and the World Media Directory.

[20] Professor Bell compiles the Pink Slime list from a NewsGuard tracker produced from a NYT subpoena, a list of pink slime networks from the Tow Center for Digital Journalism, and a list of sites identified by the parties in this litigation relating to pink slime.

[redacted]

system. Dr. Goldstein's grounding analysis answers an incorrect question for this setting. Given that a conversation may contain many grounding URLs irrelevant to news, counting the entire conversation as a "hit" whenever a single URL from the Bell News List or Bell Pink Slime List appears in the [redacted] field does not show the actual prevalence of those domains in the grounding data. To assess the actual prevalence of news domains in the grounding data, by using a URL-level denominator, I find that Bell News List hits account for 4.20% of grounding URLs (10,417 URLs); Bell Pink Slime List hits account for 0.4% of grounding URLs (1,001 URLs); Plaintiff-only Domain[21] hits account for 0.07% of grounding URLs (183 URLs), and Plaintiffs' Asserted Works[22] account for 0.004% of grounding URLs (10 URLs). These percentages are significantly lower than the figures reported in Dr. Goldstein's rebuttal analysis of 17.94% and 2.53% for the Bell News List and Bell Pink Slime List, respectively.

30. Second, Dr. Goldstein's grounding data "leak" analysis does not establish regurgitation and instead answers a fundamentally flawed question. Dr. Goldstein defines shared text in the grounding and user output fields as a "leak," claiming that this must be the result of regurgitation. Instead, shared text in the grounding and user output fields is expected as part of the normal output grounding process. Putting aside this misconception of the grounding mechanism, Dr. Goldstein's analysis is still methodologically not answering the correct question as it is filtered only to the publisher domains rather than to Plaintiffs' Asserted Works. To correct for this, I perform an analysis of shared text

---

[21] Plaintiff Domains include nytimes.com, nyt.com, nytcooking.com, cookingnytimes.com, cookingnyt.com, nytimescooking.com, cooking.nytimes.com, newyorktimescooking.com, cookingnewyorktimes.com, theathletic.com, wirecutter.com, nydailynews.com, sun-sentinel.com, chicagotribune.com, orlandosentinel.com, mercurynews.com, denverpost.com, ocregister.com, twincities.com, motherjones.com, revealnews.org, theintercept.com.

[22] "Plaintiffs' Asserted Works" refers to the specific articles that Plaintiffs have identified as the copyrighted works at issue in this litigation, as those works are defined and analyzed by Plaintiffs' technical experts. For the News Plaintiffs, the Asserted Works are the registered articles identified by The New York Times and the Daily News Plaintiffs and used in the Goldstein analysis. *See* Hurst Decl. Ex. 231, Goldstein Opening Report, ¶¶ 220–221, 225. For the CIR Plaintiffs, the Asserted Works are the registered articles identified by The Center for Investigative Reporting and used in the Wenger analysis. *See* Hurst Decl. Ex. 246, Wenger Opening Report § III.C.1. My use of this term for purposes of rebuttal adopts Plaintiffs' experts' own identification of the works at issue and does not reflect any opinion on the validity, ownership, or registration of any work.

[redacted]

between the grounding and output fields filtering for the Asserted Works, resulting in only 5,284 hits, which is 0.064% of all conversations in the 8.2 million subset of Copilot conversations containing Plaintiff keyword or domains. This is significantly lower than the results of Dr. Goldstein's analysis, which found approximately 20% of all conversations involve a "leak".

31. I also include additional responses to the rebuttal analyses of Dr. Wenger and Class Plaintiffs' expert Dr. Ben Y. Zhao, discussing the academic literature that Plaintiffs' Experts rely on to support their unfounded claims that generalization requires memorization and that training a language model is equivalent to compressing the training data. The literature which Plaintiffs' Experts rely on to support these claims uses theoretical models whose simplified assumptions do not hold for modern LLMs; the flaw in Plaintiffs' Experts' analyses is the extrapolation of those narrow results to broad conclusions about LLMs.

32. Finally, as described in Section XV, in his supplemental analysis, Dr. Goldstein provides "Illustrative Examples from Copilot Output Logs," which he claims show the nature of Copilot's copying of News Plaintiffs' content, and how the copied content also ended up in the output shown to a user and show extensive copying of the Asserted Works both in the internal grounding fields as well as the generated output shown to a user. These descriptions mischaracterize the normal behavior of output grounding, in which content is retrieved and used in conjunction with the model's pre-existing parameters to generate new text when generating responses. Dr. Goldstein's supplemental text-matching analysis adopts a more conservative approach that aligns with and confirms my corrected analysis, showing that the presence of News Plaintiffs' Asserted Works in the Copilot output and grounding fields is exceedingly rare.

IV. WHAT ARE LARGE LANGUAGE MODELS AND HOW ARE THEY CREATED?

33. As I explain below, training a large language model is a process in which text is broken into subword units called tokens, which are then converted into

[redacted]

numerical vectors and processed through a neural network that is composed of many layers. Through this process, the model learns statistical patterns of language (e.g., syntax, semantics, and factual associations) across the full breadth of its training data. The result of training is not a repository of the texts used to train the model; it is a set of billions or even trillions of numerical parameters (mathematical weights) that encode generalized statistical relationships derived from the entire training corpus.

34. LLM training requires an enormous volume and a wide diversity of textual data because the fundamental objective of training is generalization––producing a model capable of responding to new, previously unseen inputs across an effectively unlimited range of contexts and tasks. The capacity to generalize is an emergent property of learning across the complete distribution of training data, and no individual work or subset of works is responsible for conferring that capability.

35. Models can at times generate text that was part of their training corpus, a known phenomenon called regurgitation. However, instances of regurgitation in a properly trained model are extremely rare. A model that has a tendency to regurgitate its training data reflects an unintended side-effect of the training process rather than the intended operation of the model, as I discuss in more detail below. The technical concepts discussed below, including the foundations of machine learning and deep learning, neural network architectures, the tokenization and embedding processes that convert text into numerical representations, and the probabilistic mechanisms of output generation, together provide the necessary scientific framework for evaluating Plaintiffs' claims and for the opinions set forth throughout the remainder of this declaration.

A. Overview of Artificial Intelligence Technology

36. Artificial intelligence ("AI") is a field that studies how to design and build computer systems capable of performing complex tasks typically associated with human intelligence, such as reasoning, problem-solving, visual perception, and understanding language. Over time, advances in areas such as machine learning

[redacted]

and deep learning have shifted the focus from fixed programming rules to systems that identify and model patterns in data from real-world contexts. This progression from explicit rules to learned statistical representations is central to the research and development of AI models and architectures, which greatly contributed to the emergence of large language models. The discussion below places large language models within that broader technical progression, starting with the relationship between artificial intelligence, machine learning, and deep learning.

1. Artificial Intelligence, Machine Learning, Deep Learning

37. Traditional computer programs work by following instructions written by humans.[23] A program is composed of algorithms, which can be thought of as step-by-step instructions in the form of computer code, which specify how the computer processes information in order to solve some well-defined problem.[24] For example, a basic calculator may have a program that contains computer code for algorithms that perform various arithmetic operations, another that stores results for future calculations, and another that manages how numbers are displayed on the screen. Together with the hardware components such as circuitry, input buttons, and the visual screen, this forms a simple, but complete computer system for performing arithmetic. Since the invention of computers and programming, this explicit programming model has proven reliable for tasks and problems that are well-defined and structured, but it limits the capabilities of a traditional program to what programmers can anticipate and encode in advance. When a new situation arises or a different kind of problem is presented, without explicit reprogramming, the program lacks the ability to adapt.

38. AI originated within the field of computer science as an effort to build computer systems that can perform complex tasks without explicit programming by

[23] Nisan, N., & Schocken, S. (2005). The Elements of Computing Systems: Building a Modern Computer from First Principles (1st MIT Press paperback edition). MIT. pp. 57–62.
[24] Computer Programming. (n.d.). National Library of Medicine. Retrieved January 9, 2026, from https://www.nnlm.gov/guides/data-glossary/computer-programming.

[redacted]

humans for each new problem. AI research examines how a computing machine can perceive its environment, interpret its observations, and make decisions or predictions to achieve a specific goal or solution to a problem. The practical focus of modern AI is on designing systems that can improve their performance on defined tasks through exposure to data, a capability that distinguishes AI from traditional programming.

39. The first generation of AI systems emerged in the 1950s and 1960s, as researchers attempted to reproduce reasoning and problem solving through symbolic logic and rule-based control. While these programs could perform specific tasks, their "intelligence" depended entirely on instructions and knowledge explicitly encoded by humans. As AI progressed further in the 1970s, expert systems applied similar ideas on a larger scale by storing specialized knowledge in extensive rule sets.[25] But these systems remained rigid and unable to learn from new data. The narrow scope showed the limitations of rule-based methods and motivated the development of approaches capable of learning directly from experience, a transition that would define the next stage of AI.[26]

40. Machine learning ("ML") is a subfield of artificial intelligence that studies how computers can perform tasks that improve with experience and adapt to new data, without the use of fixed, pre-programmed rules. Data captures rich information about systems and environments and, although often complex and diverse, typically contains underlying structure that can be represented as patterns. ML models are designed and built to detect these recurring trends or statistical regularities within data. For example, specific colors or shapes may occur frequently in particular types of images, customers who share similar purchasing habits may make related buying decisions, and certain words may often appear together in sentences. Detecting such patterns allows ML systems to

[25] Russell, S. J., et al. (2022). Artificial Intelligence: A Modern Approach (Fourth edition, global edition). Pearson. pp. 36–41.
[26] Sutton, R. (2019, March 13). The Bitter Lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

[redacted]

model relationships within the data and to apply those learned relationships to new, previously unseen inputs. This capability is known as generalization.

41. Machine learning systems require large amounts of data during training in order to detect and represent the statistical patterns described above. The more complex and varied the domain (and natural language is among the most complex), the greater the volume and diversity of data needed for a model to learn patterns that generalize reliably to new inputs. By processing many examples of the data via "training," the model learns how different features of the data tend to vary together, forms internal representations of those connections, and can recognize similar structures in new data. In this framework, patterns describe regularities in data while relationships characterize how one variable changes in association with another. For instance, a model that predicts house prices may detect a pattern showing that houses within a neighborhood share similar price ranges, and a relationship indicating that prices typically increase with square footage. Modern machine learning systems, built on these principles, power a wide range of applications such as fraud and spam detection, facial recognition, voice recognition, and recommender systems for movies, products, and other media. These models can address complex, real-world problems, with limited explicit programming, and can achieve higher performance when exposed to larger and more diverse data.

42. Deep learning ("DL") is a subfield of machine learning that has driven many of the recent advances seen in AI and is fundamentally based on a particular class of machine learning models known as neural networks.[27] A neural network is a network-based architecture composed of layers of interconnected computational units. Each layer transforms the input it receives, and extracts features that become increasingly abstract at greater depth. The successive transformations of data across these layers enable deep learning systems to process and represent

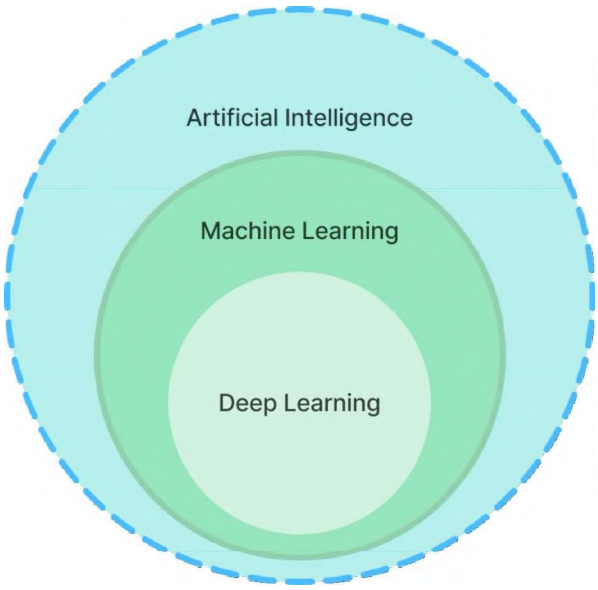
[27] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. pp. 1-2.

[redacted]

complex data such as images, speech, and text, as numerical representations that capture the statistical structure of the input.

43. In Figure 1 I show the relationship between artificial intelligence, machine learning, and deep learning.

Figure 1: Comparison of Artificial Intelligence, Machine Learning, and Deep Learning



44. Beyond data, advances in computational resources and infrastructure, commonly known as "compute," have been central in the rise of modern deep learning systems. Compute encompasses the hardware and processing capacity that perform the intensive numerical operations and calculations required for building and running deep learning models. The development of high-performance hardware such as graphics processing units (GPUs) and other specialized chips has significantly increased the speed and scale of computation. Compute also includes the supporting infrastructure that manages memory, data transfer, and storage across distributed computer systems that facilitate model development. The scale and capability of a deep learning model are a function of all three inputs (data, model architecture, and compute) working in combination; no single input is sufficient on its own to produce a capable model.

[redacted]

45. Advances in data, model design, and compute shaped the development of deep learning across several domains. The first major successes for deep learning were in the domain of image classification of natural images.[28] In the domain of processing natural language, deep learning models learn directly from large and diverse text collections, rather than relying on fixed grammatical rules written by developers, which are limited in their ability to handle arbitrary text. Natural language is among the most complex and variable forms of data: it spans an effectively unlimited range of topics, styles, and contexts, and its meaning depends on relationships among words that operate at multiple levels. Training neural network language models on this domain therefore requires training on an extensive amount of online, digital text, so that neural network language models learn to identify patterns in language that implicitly represent grammar, semantics, and context across the entirety of their training data. These learned statistical patterns do not represent the content of any individual text or collection of texts; they are emergent properties of the complete distribution of training data. The convergence of developments in data, computation, and model design provide the essential ingredients for today's large language models ("LLMs") –– a general-purpose technology capable of generating, analyzing, and transforming text across a wide and open-ended range of tasks and subject matters.

2. *Learning Paradigms*

46. Throughout science, the term "model" is used to refer to a simplified representation of a system or natural phenomenon, used to explain, analyze, or predict aspects of the system or phenomenon of interest. Within AI, and ML in particular, a model refers to a computational structure that is built and refined through machine learning algorithms. On a functional level, these models process

[28] Krizhevsky, A., et al. (2012). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60 (6), 84–90. https://doi.org/10.1145/3065386.

[redacted]

inputs and produce outputs. Associations between input patterns and outputs are formed during model "training."

47. Inputs are data that the model receives, and outputs are the results generated from processing those data. In the context of AI models, inputs often consist of numerical representations of text, speech, images, or other observations of a real-world environment or phenomenon. The outputs correspond to predictions derived from those inputs, and their representation and meaning largely depend on the model's specific objective and design.

48. A machine learning model must undergo training to interpret input data and develop an ability to produce meaningful outputs and accurate predictions. Data is central to this process because the model improves only when it receives feedback, either explicit or implicit, that distinguishes accurate predictions from errors. However, data alone is not sufficient for learning if the model cannot determine whether its predictions are correct during training. For example, a system designed to identify fraudulent emails (spam) may not be able to improve if it is provided a mixed collection of messages, without knowing which examples are legitimate and which are not. Such feedback is used to allow the model to "learn" and improve its predictions.

49. To address this problem, developers either supplement the data with informative signals, such as labels, or redefine how the model learns directly from the data itself. In machine learning, a label is an attribute or category assigned to a data point that represents what the model is expected to predict. In the spam detection example, emails in the dataset may be labeled as "spam" or "not spam." For image classification, labels might indicate categories such as "cat," "dog," or "bird." By associating input data with labels, the model can compare each prediction or output, against the desired output during training. Labeling is one approach, but researchers also design techniques that redefine how a model interacts with data, without explicit labels. These strategies or "learning paradigms" most commonly include supervised learning, unsupervised learning, self-supervised learning, and reinforcement learning. Developers choose among

[redacted]

these approaches based on the type of data available, the purpose of the model, and the nature of the task.

50. In supervised learning, a model is trained on data for which the desired outputs are known. Each example in the dataset includes both the input and its corresponding labeled output, and the model's objective is to capture the relationship between them. For example, an image recognition system may be given thousands of images of animals (the input), each labeled with the category corresponding to the animal depicted, such as "dog," "cat," or "bird" (the output). This is often framed as a prediction problem by requiring the model to output a probability distribution over the possible labels, with the distribution changing for each data point. As more data are processed by the model, and with increased computation time, the model identifies features and patterns that distinguish among animal types to make accurate predictions. Once the model has captured these relationships, it can classify new images of animals. This is usually done by assigning the labels that have the greatest probability under the model. Supervised learning is widely used in applications such as handwriting recognition, where datasets include labeled examples or known outcomes. The main advantage of this paradigm is its ability to produce accurate models for well-defined tasks where sufficient labeled data exists. However, it is limited by the time, resources, and cost required to create such labeled datasets, and the ability to determine a well-defined outcome.

51. In unsupervised learning, the system is provided data without labels and aims to discover patterns or groupings from the data itself. For instance, a music analysis system may be given thousands of songs (the input) without categories or any other label that classifies them. As the unsupervised learning system processes the songs, it may cluster them by attributes such as tempo, instrumentation, or style, eventually forming groups that correspond to genres without prior knowledge of the concept of a genre. This paradigm is useful particularly when the categories are not known or defined in advance, or when labeling the data would be prohibitively expensive or time-consuming. As there are no explicit

[redacted]

"correct" answers, performance is evaluated by how effectively the system discovers meaningful patterns and structure in the data.

52. Self-supervised learning (the paradigm used to pre-train large language models) refers to a particular type of unsupervised learning, where the data are broken up into "pieces" and the model is trained to predict one piece at a time. The "pieces" are designed to be basic, and could be small regions of an image, or individual words in a corpus of text. This type of machine learning is important in generative AI because the model learns to predict or "generate" the training data and can then synthesize new data that have not been observed before. In essence, self-supervised learning recasts unsupervised learning as a series of supervised learning problems, where the "pieces" correspond to the labels. Returning to the music example, a model might predict the next sequence of musical notes based on the past, by implicitly capturing the style of music without directly clustering songs. This approach is well suited to language because, for any given text, a portion of it can be withheld to serve as the label, and the task of the model is to predict the missing element. For example, given a sentence "Winter in Siberia is very cold" the system may modify the phrase into "Winter in Siberia is very " where "cold" serves as the label. The system then predicts the missing word using information from the surrounding text, such as "Winter" and "Siberia." This method can be applied at scale to raw, unlabeled data, including books, articles, social media posts, and even computer source code. It is this property that both enables and necessitates training on vast and diverse text collections. The more varied the language the model is exposed to, the more robustly it learns the statistical patterns (structural and semantic properties of language) needed to make accurate predictions across new and unseen contexts. The essence of self-supervised learning is that if the model is able to consistently and accurately predict the missing word, the model must have implicitly captured important aspects of these linguistic structure and semantic relationships (inherently a generalization objective), accomplished without the need for expensive, expert human annotation. Importantly, prediction in self-supervised learning can be quantitatively specified as a mathematical "loss function" (quantifying the

[redacted]

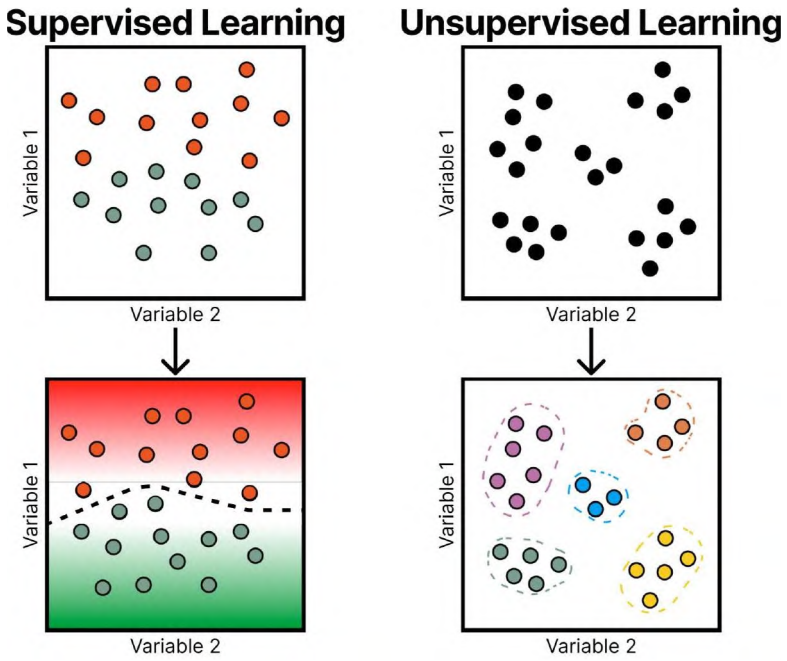
difference between the model's predictions and the actual data) that the training process seeks to optimize. I discuss the optimization of the loss function further in Section IV.A.5.

53. Figure 2 compares unsupervised and supervised learning. On the left-hand side labeled, "Supervised learning," the top graph depicts labeled data points corresponding to red and green colors plotted along two variables. Below that, the graph shows a boundary learned by the model that separates the two classes of labeled data points. On the right-hand side labeled, "Unsupervised learning," the top graph depicts unlabeled data points plotted along two variables. Below that, the variables are clustered into five groups using an unsupervised algorithm. While this example shows unsupervised learning as a form of clustering, in general, and in the case of LLMs as I explain below, many overlapping patterns are identified without leading to explicit clusters of data points.

54. Reinforcement learning ("RL") is a machine learning paradigm that is based on models of interaction between an agent (an entity that acts to achieve long-term goals) and an environment. In other words, a model learns to make sequential decisions by receiving feedback in the form of rewards or penalties based on the outcomes of its actions, rather than from labeled examples. The actions the agent takes change the state of the environment, and the agent receives rewards for each action. The goal of the agent is to maximize the cumulative reward received. In this way, RL is a framework for sequential decision making to achieve a long-term goal. RL is not used for LLM pre-training, but a variant known as reinforcement learning with human feedback (RLHF) plays an important role in the post-training alignment of language models, as I discuss in Section IV.D.

[redacted]

Figure 2: Example of Supervised and Unsupervised Learning Paradigms



55. These machine learning paradigms are implemented using a variety of model architectures. Among these, neural networks are the most important building blocks used by modern language models.

3. Neural Networks

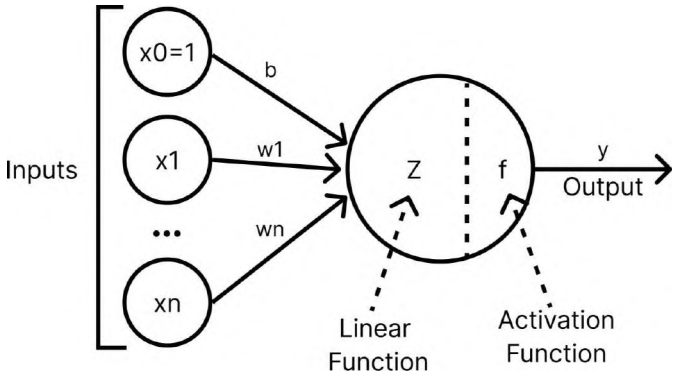
56. Neural networks, a type of machine learning architecture, form the foundation of deep learning and enable modern language models to process and analyze input text, and generate novel content. A neural network is a computational structure composed of layers of interconnected processing units (called neurons) that apply a series of mathematical operations to input data. Each layer transforms the numerical data it receives and passes the result to the next layer. Through these successive transformations, the network converts raw input data, such as text, images, or audio, into numerical representations that capture increasingly complex statistical patterns. The network's capacity to model such patterns is determined by its architecture: the number of layers, the number of neurons in each layer, and the mathematical operations applied at each stage. This section explains the basic structure of neural networks, the role of neurons and layers,

[redacted]

and the development of sequence-based architectures that preceded transformers.

57. At the core of every artificial neural network is a fundamental computational unit called a neuron.[29] A neuron receives numerical inputs, computes a weighted sum of those inputs, and passes that weighted sum to a nonlinear function which produces a new number, which is the output of the neuron.[30] The weight assigned to each input determines how much influence that input has on the neuron's computation. A large weight amplifies the contribution of a particular input; a small weight diminishes it. In a neural network, each neuron receives inputs from several other neurons, and the output computed by the neuron from those inputs is then sent as new input to other neurons in the network. Conceptually, a neuron functions as a small decision unit: it receives pieces of information from other neurons as inputs, assigns them importance based on the strength of the connection with the incoming neuron, and generates an output representing the strength of its response. An individual neuron can represent only simple relationships, but within a large network, these neurons can collectively model highly complex patterns.

Figure 3: Example of Neuron



58. Figure 3 illustrates this in more detail, "x1" and "xn" represent the neuron's inputs, "w1" and "wn" represent the corresponding weights, "Z" represents the

[29] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. p. 16.
[30] Russell, S. J., et al. (2022). Artificial Intelligence: A Modern Approach (Fourth edition, global edition). Pearson. pp. 802–803.

[redacted]

weighted sum (a linear combination, or weighted sum, of the inputs). Linear combinations alone are limited, however, because they can represent only simple relationships among variables, whereas most real-world data involve non-linear or irregular dependencies. To model such complexity, the result of the weighted sum is passed through a non-linear activation function, represented by "f" in Figure 3.[31] This function determines whether the neuron becomes active and how strongly it contributes to the next stage of computation. The neuron's output, represented by "y" in Figure 3, becomes an input value for other neurons for subsequent processing.

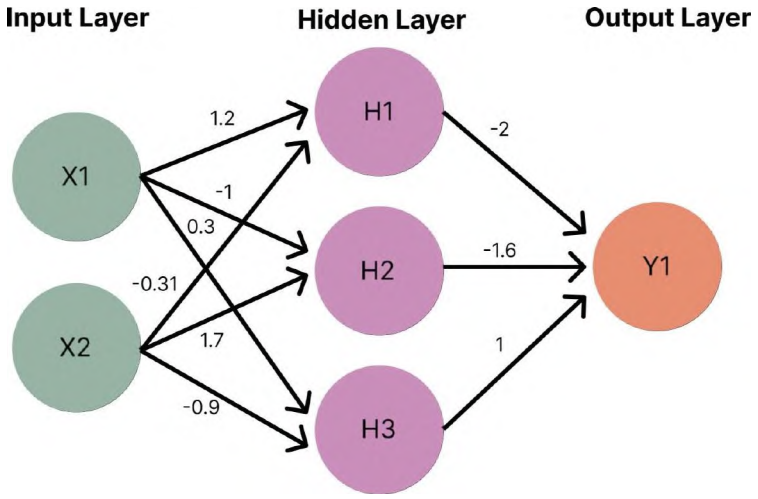
59. Neurons are organized in interconnected layers that form a network, which I have demonstrated below in Figure 4. The first layer, known as the input layer, contains neurons that receive the raw numerical data. Subsequent layers, commonly known as hidden layers, perform additional transformations on the data and pass the results forward to the next layer.[32] The final layer performs the last set of transformations before producing the network's prediction or output. The intermediate computations carried out in the hidden layers are not directly visible to the end user. What the user sees is the input provided to the system and the output it ultimately produces, not the many internal numerical transformations that occur between them.

60. An example of a neural network is a handwritten-digit classifier: the input layer receives the pixel values from an image of a handwritten digit, the hidden layers detect localized patterns such as curves or lines, and the output layer identifies what digit the image represents. The structure of the neural network is determined by such properties as the number of layers and the number of neurons in each layer. These properties are a design choice that influences the ability of the network to identify both simple and complex patterns in the data.

[31] Sharma, S., et al. (2020). Activation Functions In Neural Networks. International Journal of Engineering Applied Sciences and Technology, 04 (12), 310–316. https://doi.org/10.33564/IJEAST.2020.v04i12.054.
[32] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. p. 169.

[redacted]

Figure 4: Example of a Neural Network



61. The depth of a neural network refers to the number of layers within its architecture and is a primary determinant of the complexity of patterns the network can represent. A shallow network with a few layers may only capture simple relationships in the data and will struggle with more complex tasks. Adding hidden layers generally allows the network to build hierarchies of features, in which each layer draws on the representations formed in previous layers. The optimal size of the network typically depends on the scale of data; modern deep learning systems used to process extremely large data collections often have hundreds of layers. As a natural result of the training process, these layers structure themselves, which allows for the emergence of hierarchies of features. This hierarchical organization, from simple patterns to more abstract representations, allows deep networks to model data with rich and complex structure.

62. The neural networks described so far are called feed-forward neural networks, also called multilayer perceptrons, which process each input independently, with information flowing in a single direction from input to output.[33] Language, however, is inherently sequential and occurs as sequences that unfold over time, where order and temporal position are key properties. To handle such data,

[33] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. p. 168-169.

[redacted]

researchers developed sequence-based neural networks such as Recurrent Neural Networks ("RNNs") and Long Short-Term Memory ("LSTM") networks, which were designed to maintain internal states that carry information from earlier inputs forward, enabling the model to account for context.[34] This property makes them suitable for tasks that depend on temporal relationships, such as speech recognition, language modeling, and other sequential prediction tasks.[35] At each step in the sequence, the network updates its internal memory so that it represents both the current input and the accumulated context. While these models expanded the ability of neural networks to represent time-dependent data, they struggled to retain information across very long sequences.[36] These limitations motivated the development of new models that could manage long-range dependencies in sequences, leading to the emergence of transformer-based architectures, which are discussed in Section IV.B.6).

4. Models and Parameters

63. Earlier in Section IV.A.1, I describe how, in some scientific and engineering contexts, a model is a representation of a system, process, or phenomenon used to describe or predict behavior. Traditional scientific models may have a small number of adjustable parameters that have physical significance, such as the viscosity of a fluid when modeling its flow through a tube. In AI, and specifically in ML, a model is a computational system that takes in inputs and produces outputs using a set of adjustable values known as trainable parameters. These parameters determine how the model processes information and are what enable it to improve with experience. Unlike traditional programs and algorithms, which always produce the same output for a given input based on fixed instructions, a machine learning model can adapt over time by adjusting its trainable parameters in response to data, thereby refining how it interprets and represents

[34] Russell, S. J., et al. (2022). Artificial Intelligence: A Modern Approach (Fourth edition, global edition). Pearson. pp. 823–824.
[35] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. pp. 373–374.
[36] Russell, S. J., et al. (2022). Artificial Intelligence: A Modern Approach (Fourth edition, global edition). Pearson. pp. 914–915.

[redacted]

patterns. This section explains what model parameters are, how they are adjusted during training, and what trained parameters represent once training is complete.

64. Trainable parameters are numerical values within a model that are automatically refined during the training process and determine how the model responds to different aspects of the input data. The most common and important trainable parameter in a neural network is a weight, which scales an output of a neuron before it is passed forward through the network to another neuron. A weight represents the amount of influence that the first neuron has on the second. Another trainable parameter, the bias, determines an offset applied to the collection of inputs to a neuron. A bias parameter represents how large the input signal to the neuron must be in order for the neuron to become activated. Both types of parameters –– weights and biases, together determine how the signal moves through the layers of the network and how much influence the model gives to each feature of the input. By contrast, hyperparameters are settings chosen by developers before training, such as learning rate, batch size, or number of layers, and are not themselves learned from the training data.[37] For purposes of this declaration, the term parameters refers to trainable parameters unless otherwise stated. Adjusting the parameters, which make up the internal structure of the model, also changes the function of the model as it transforms how the model interprets the inputs and generates a prediction.

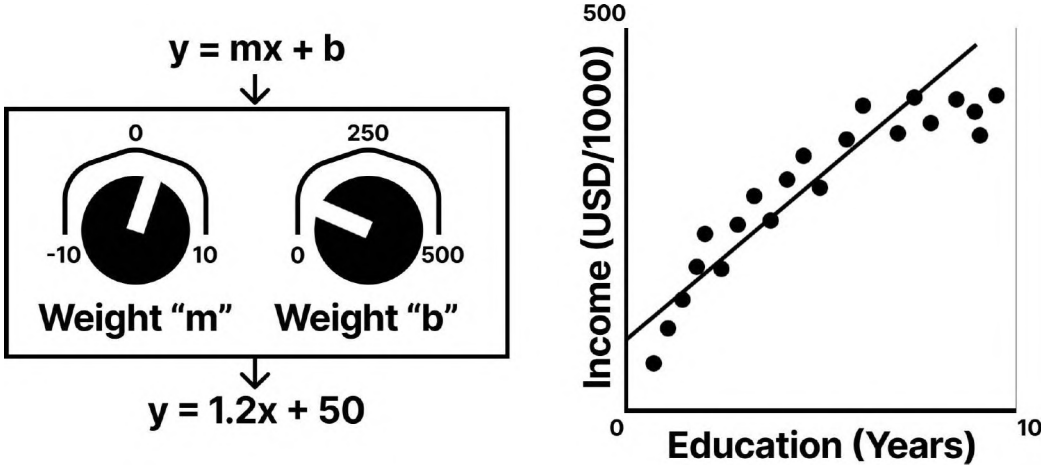
65. During model training, the parameters are not fixed, but rather are updated iteratively to improve the model's ability to make accurate predictions. A model passes an input to generate an output, which in turn is compared to the expected output to calculate an error signal. The optimization algorithm uses this error signal to adjust the parameters so the error is reduced. Through repeated updates over many training examples, the model learns and reduces its overall error based on the statistical regularities across the data. The process will result in a set

[37] Goodfellow, I., et al. (2016). Deep Learning . The MIT Press. p. 98.

[redacted]

of parameters that have learned generalizable patterns across the entire distribution of data it is trained on, and not the memorization of specific examples. I provide an example of model parameters tuned to adjust the slope and intercept of a straight line to best fit data on income vs. education in Figure 5. During training, these knobs are automatically adjusted by the optimization algorithm to reduce the total distance between the points on the plot and the line. Estimating model parameters of a neural network involves a similar process, but there can be billions of parameters, which collectively determine the relationship between inputs and outputs. After training, these parameters will become set in place.

Figure 5: Model Parameters



66. A model's parameters (the billions of numerical weights and biases discussed above) are not designed to store, contain, or represent any specific piece of training data, nor any individual text, image, or other training example. Rather, they constitute a mathematical representation of the statistical patterns (relationships, structures, and dependencies) that the training process extracted from the aggregate of all training data. For example, in language models as discussed in Section IV.B. below, the parameters implicitly reflect the syntactic and semantic tendencies and features of language itself, specifically how words, phrases, and ideas relate to each other, all without explicitly storing individual sentences from the training data in memory. This enables a model to generalize to new contexts beyond its training data while still creating response outputs that

[redacted]

are coherent and relevant by applying the statistical patterns it has learned, patterns that reflect the properties of language as observed across the training data as a whole.

5. Training, Testing, and Evaluation

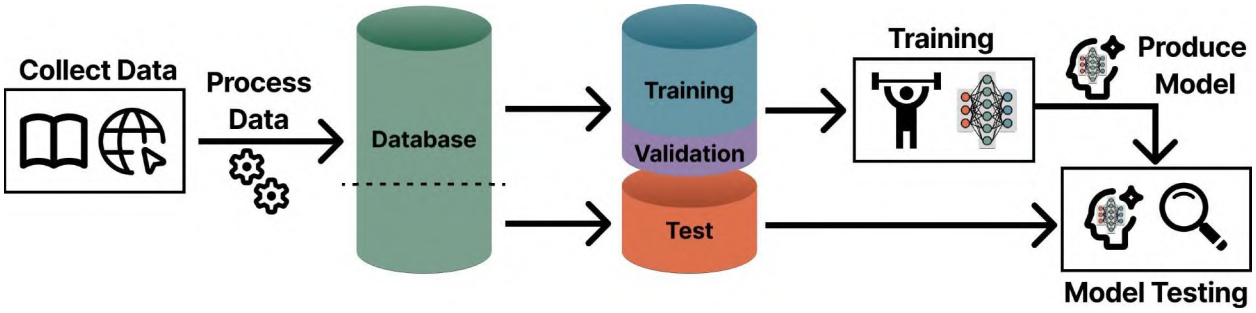
67. Developing a machine learning model involves three interrelated stages: preparing and partitioning data, training the model on a portion of that data, and then testing and evaluating the trained model's performance on data it has not previously encountered. The process of training, testing, and evaluation is a critical 3-step process that is generally used to develop and refine machine learning models.

68. Training describes the phase in which the model adjusts its parameters to learn from data, by reducing prediction error on the training data as described above. In contrast, testing and evaluation measure how well the trained model performs when given new observations. Together, these stages not only determine whether a model can perform a task but also if the model can generalize by applying what it has learned beyond the examples it was trained on. This section describes how datasets are prepared for those stages, how models are trained and tested, and how performance is evaluated to determine whether the model generalizes beyond the examples on which it was trained.

69. The first stage of developing a machine learning model is assembling and preparing the training data. Raw data such as text, images, or other digital records must first be collected, filtered, and formatted into a standardized structure that is suitable for computation. This may involve removing duplicate data, correcting errors, normalizing inconsistent formatting, and filtering out low-quality or irrelevant material. In the context of language model development, this stage involves obtaining large volumes of text from diverse sources and processing that text into a form that can be used as input to the training algorithm. Data acquisition and preparation is not a standalone activity separate from model development; it is a technical prerequisite without which training cannot proceed, and the design choices made at this stage (what data to include,

how to filter and clean it, how to structure it) directly affect the statistical patterns the model will learn. Once processed, the curated corpus is divided into three distinct subsets, which I have illustrated in Figure 6. The first is called a training set, which the model uses to tune its parameters. Next is a validation set, which is used to monitor the model's current performance during the training phase. Finally, a test set is held out entirely until the model's final evaluation. This separation is important in the overall design and process to assess and measure the model's performance accurately. In particular, this pipeline is used to reliably compare multiple models, often with different choices of hyperparameters, in order to select the best one.

Figure 6: Example of Train/Test Data Split



70. When a model is being trained, it is presented with large amounts of input data and incrementally changes its parameters, little by little, in order to lessen the error. In practice, the model is given data in groups or batches, and for each batch it makes a set of predictions, which are then matched against the actual, desired outputs. The difference between the predicted outcome and actual result, commonly referred to as the error or loss, is the basis for determining how the parameters of the model should be adjusted. Through optimization algorithms, small changes are made such that the overall error in each batch is reduced. One complete pass through the full training dataset is commonly referred to as an epoch. Training may involve multiple epochs, where the model processes the same corpus more than once as its parameters are updated. This continues for many passes through the training data as the parameters are gradually adjusted so that the model determines patterns and features of the examples it has encountered that are predictive of the desired outputs.

[redacted]

71. As training epochs continue, the generalization ability of the model is evaluated through testing. In this phase, the model will be presented with data it has not seen during training. Since the model does not know the correct answers for this data, testing provides a more objective measure of performance. Training continues as long as the model's performance on the unseen data continues to improve. Once the model begins to perform well on training data but begins to lose accuracy on testing data, the training stops. At that point, if training were to continue, the model would "overfit"; in this case it would begin to use its weights to memorize the training examples rather than to learn more general principles. Similarly, if it performs poorly on both the training and testing data, it has underfit; in this case it did not learn sufficient structure from the data. The overfitting problem, and the techniques developers use to prevent it, is discussed in more detail in Section IV.A.6).

72. After completing the training, the weights of the model contain billions of mathematical relationships gleaned from the structure of the training data structure but is not constrained to recreate it. Properly trained models that strike a balance perform well on unseen inputs because their parameters capture general patterns rather than specific cases. Evaluation is the broader process of analyzing a model's performance using quantitative and qualitative indicators. Quantitatively, the evaluation entails various metrics such as accuracy, precision, recall, or loss values to measure how well a model meets defined objectives.[38] For language models, evaluation will also include domain-specific and specialized benchmarks that measure reasoning, comprehension, or ability to generate coherent and contextually relevant text.[39] Qualitative aspects of evaluation are equally important for measuring specific aspects of model behavior that might not be quantifiable such as tone, relevance, and factual reliability, and may rely on human review. Evaluations are designed to ensure the model does not solely

[38] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. pp. 422–424.
[39] Prince, S. J. D. (2023). Understanding Deep Learning. The MIT Press. p. 234.

[redacted]

produce a correct output or prediction in a mathematical sense, but also generally performs as expected in practice.

6. Overfitting and Underfitting

73. Machine learning models are designed to learn patterns that generalize outside of the set of data on which they were trained. Two complementary problems arise when this balance is not achieved: overfitting and underfitting. These problems both describe how well a model's internal parameters capture the important structure in the data rather than its incidental structure or noise. This section explains each problem and then contrasts both with the balanced fit associated with effective generalization.

74. Overfitting occurs when a model fits training data too closely, over indexing on specific sequences or patterns that are not representative of the overall population.[40] When training a model, its parameters will be adjusted as long as the error is decreasing. In the event that the dataset size is insufficiently small or when the model is trained for too long, it will record random variation and noise instead of capturing the meaningful patterns in the data. The result of this overfitting is a high accuracy on the training set, but a poor performance on the test data set. This is due to the test data set having examples of data that were not fit in the training stage, leading to poor generalization. An overfit model may be capable of re-creating sequences it has seen before in training data but is unable to adjust to generalization. Overfitting is a well-understood failure mode, and developers employ a range of techniques specifically designed to prevent it. These include limiting training duration, occasionally setting some neurons or connections to zero (referred to as "dropout"), and using different regularization techniques. The overfitting problem is related to both the volume and diversity of the training data, and the complexity of the model. A model that is overly complex, with a large number of tunable parameters, is more likely to overfit on a small data set that lacks diversity.

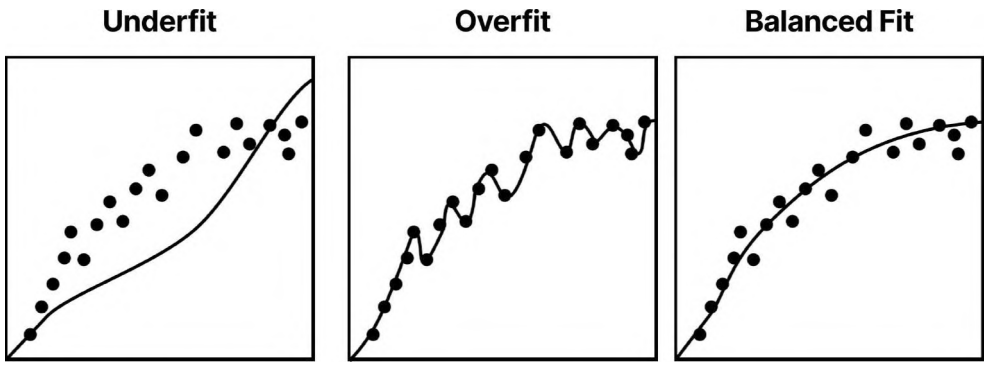
[40] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. pp. 111–113.

[redacted]

75. Underfitting represents the opposite problem where the model fails to capture enough of the structure in the training data to make accurate predictions.[41] This can occur when the model's architecture is too simple for the task, when it has insufficient data to learn from, or when training is stopped before the model has fully converged. An underfit model performs poorly both on the data it has seen and on new examples because it has not learned the underlying relationships. In practice, underfitting produces models that are biased toward overly simple explanations and fail to recognize the complexity of patterns in natural language, images, or other forms of data.

76. A model with a balanced fit captures important structure in the data while still being able to generalize to data that was not seen during its training. In Figure 7 below, I provide an example of how these types of fits may appear on a two-dimensional plane. The goal of the model development process, including the data preparation, architectural design, and training techniques described throughout this section, is to produce a model in the "Balanced Fit" state, one whose parameters encode generalizable patterns of language rather than the content of specific training examples.

Figure7: Example of Model Fit



B. Large Language Model (LLM) Pre-training

77. The preceding sections described the general principles of machine learning, deep learning, and neural network architectures. This section applies those

[41] Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. pp. 111–113.

[redacted]

concepts to the specific class of models relevant to Plaintiffs' allegations: large language models. As introduced above, an LLM is a deep neural network trained using self-supervised learning on vast quantities of text, with the objective of predicting the next token in a sequence. The term "pre-training" refers to this initial, large-scale training phase, and is used to describe the process of tuning the parameters of a large language model, which is a specialized type of deep neural network. The parameters are optimized using a simple but powerful objective: to predict, one-by-one, the words (or "tokens", discussed below in Section IV.B.6) in a large corpus of text. Through the pre-training process, the parameters of the neural network represent a vast amount of generalized information about the semantic structure, grammar, style, and other types of word co-occurrence patterns underlying the entirety of data it was trained on. Once the LLM is fully trained after pre-training, subsequent machine learning steps are used to make the model useful for different tasks.

1. Overview and Characteristics

78. LLMs are deep learning systems designed to process and generate human language. They are a distinctive class of neural networks based on the Transformer architecture, trained on large datasets of text data using an objective that is surprisingly simple: predict the next word to appear in the sequence.[42] This section introduces the basic characteristics of LLMs, explains the significance of autoregressive next-token prediction, and distinguishes between a pre-trained LLM and later post-training steps that adapt the model for particular uses.

79. Precursors to today's LLMs have been used in search, speech recognition, and machine translation systems since the 1990s. To make search in multiple languages possible, Google developed statistical models for translation that used language models in a central way, building on early work at IBM. In a 2007

[42] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). pp. 147–148.

[redacted]

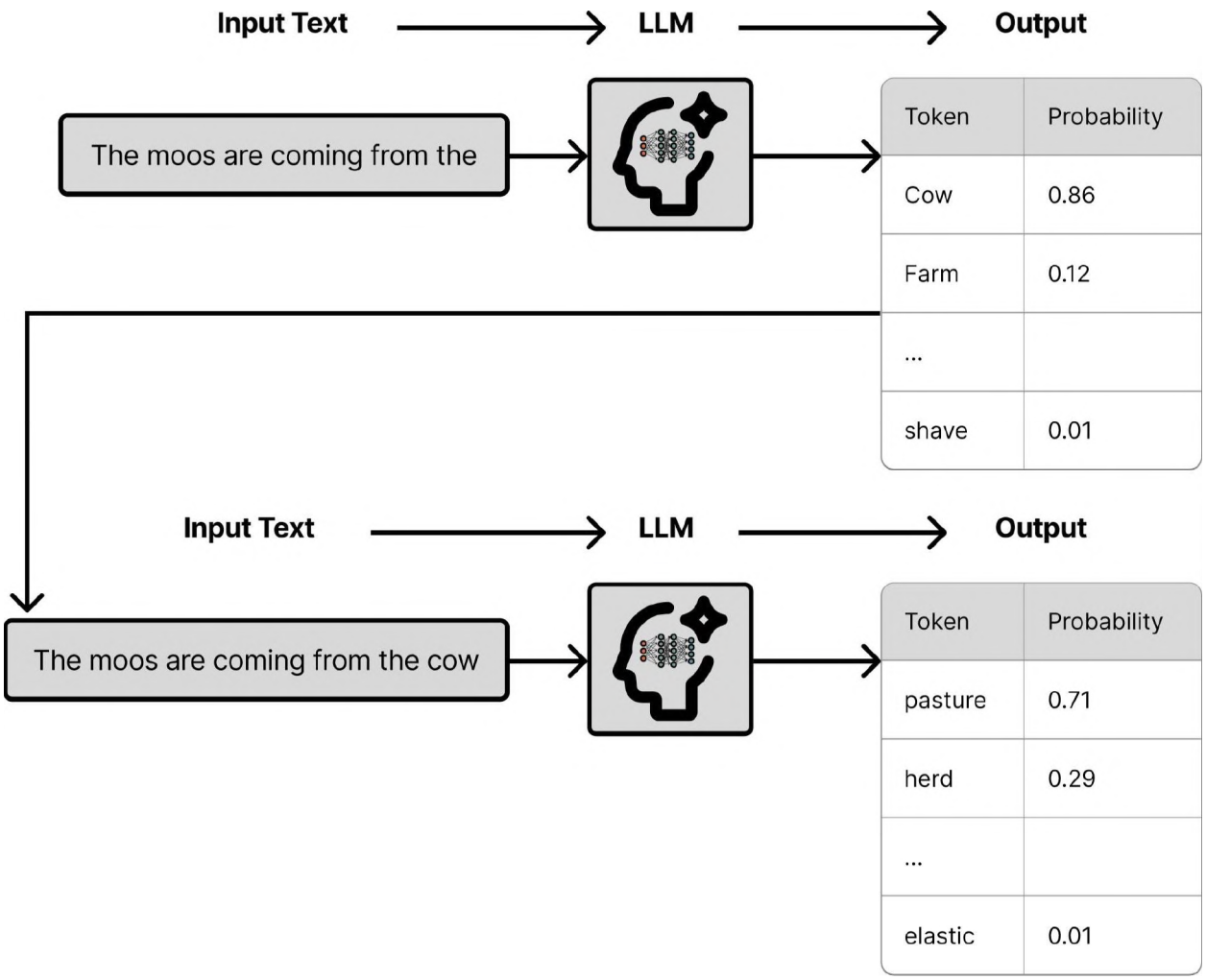
paper,[43] researchers at Google demonstrated how simply training n-gram language models on larger quantities of text gave continued improvements in translation accuracy. This work trained language models on up to 2 trillion tokens of text, collected from various sources, including news articles and general data collected from the internet in 2006, and documents some of the first "scaling laws" for large language models that were observed in the industry.[44] In the academic literature, language modeling for search and information retrieval was studied in the early 2000s.[45] The core neural network technology underlying these search systems is now driving the next major wave of innovation in LLMs. Search and LLMs use similar neural network techniques to understand language, but they employ them for different purposes. Where search engines rank existing content, LLMs generate new content.

80. Most modern LLMs are trained to predict the next token autoregressively, meaning the model predicts the tokens one-by-one, in the order they appear in the training text, and using only past tokens as a source of information for predicting future tokens. I demonstrate an example of autoregressive next token prediction in Figure 8 below, and explain the process further in Section IV.B.6).

[43] Brants, T., et al. (2007). Large Language Models in Machine Translation . https://aclanthology.org/anthology-files/anthology-files/pdf/D/D07/D07-1090.pdf.
[44] I explain scaling laws for LLMs in Section IV.B.8 below.
[45] Croft, W. B., & Lafferty, J. (Eds.). (2003). Language Modeling for Information Retrieval . Springer Netherlands. https://doi.org/10.1007/978-94-017-0171-6.

[redacted]

Figure 8: Autoregressive LLM Generation



81. In order to carry out this task well, which can be mathematically and statistically characterized, the model must capture and identify rich co-occurrence patterns of words, phrases, and concepts that underlie natural language. After being exposed to large volumes of training text, LLMs create statistical representations of the patterns of words in that training text, enabling the model to produce new text in response to entirely new settings.

82. It is useful to distinguish between the pre-trained LLM, and AI tools that are derived by post-processing steps after the LMM is trained. The pre-trained LLM learns generalized information about the patterns of language usage across the entire distribution of text it was trained on. The post-processing steps, described later in Section IV.C., further develop the model's ability to perform in specific

[redacted]

response patterns, tailored to specific contexts, and aligned to desirable characteristics such as helpfulness and safety. Each of these stages involves additional engineering, data, and design decisions that contribute independently to the functionality and value of the resulting product.

2. Applications

83. Since pre-trained LLMs are trained on diverse text and learn statistical relationships across language, they are effectively general-purpose systems that have the capability to be applied to many different tasks, without needing to be built separately for each one. That general-purpose capability is ultimately a direct consequence of training the model to generalize across many subjects, writing styles, and forms of expression. As a result, LLMs can be adapted for a wide range of useful applications which do not depend on reproducing training text.

84. Recent studies suggest that over 40% of working age adults use some form of AI for work.[46] I understand that Microsoft's other Experts are addressing the uses of Copilot through systematic methodologies. This section briefly describes several representative applications of LLMs based upon their inherent capabilities, including drafting, summarization, data analysis, tutoring, language learning, software development, scientific research, and customer-facing applications.

85. In practice, LLM-powered applications include drafting correspondence, summarizing documents, analyzing data, tutoring users on academic or professional topics, supporting language learning, and assisting with brainstorming or idea generation. For example, an LLM may help a user prepare a business email, condense a long report into a shorter summary, organize information from a dataset into a readable narrative, explain a concept step by step to a student, translate or rephrase text for a language learner, or generate alternative outlines or suggestions during early-stage writing. These uses reflect

[46] Generative AI Adoption Tracker. (n.d.). Generative AI Adoption Tracker. Retrieved March 17, 2026, from https://GenAIAdoptionTracker.com.

[redacted]

the model's ability to apply learned patterns of language to new prompts and new contexts.

86. In software companies, LLMs are used in many areas, including generating and detecting issues with code, aiding in project management, and suggesting optimal software architectures.[47] Some software companies go further, using LLMs for data science, marketing, product design, and legal purposes.[48]

87. The software industry was one of the first to widely adopt LLMs, with a 2021 paper showcasing training an LLM on publicly available code from GitHub, which was able to generate code for a variety of purposes.[49] This use case was so helpful, GitHub itself later introduced its own LLM-based product for coding, a system called GitHub Copilot.[50]

88. LLMs are also being used in the field of bioinformatics to perform tasks such as analyzing DNA, RNA, and proteins. In this domain, LLMs analyze nucleotide sequences (A, T, C, and G) and are then trained to predict the functions of genes and the structure of proteins.[51]

3. *Data Preparation and Diversity*

89. The first step of training an LLM in any application is to source and prepare the requisite training corpus comprising multiple datasets. This section describes the scale and diversity of the text used to train modern LLMs, the reasons that data diversity matters, and the preprocessing and filtering steps used to prepare the data for training.

---

[47] Downie, A., & Finio, M. (2024, October 7). AI in software development. IBM. https://www.ibm.com/think/topics/ai-in-software-development.

[48] How Anthropic teams use Claude Code. (n.d.). Anthropic. Retrieved March 18, 2026, from https://www- cdn.anthropic.com/58284b19e702b49db9302d5b6f135ad8871e7658.pdf.

[49] Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code (arXiv:2107.03374). arXiv. https://doi.org/10.48550/arXiv.2107.03374.

[50] Dohmke, T. (2022, June 21). GitHub Copilot is generally available to all developers. The GitHub Blog. https://github.blog/news-insights/product-news/github-copilot-is-generally-available-to-all-developers/.

[51] Wang, Z., et al. (2026). Large Language Models in Bioinformatics: A Survey (arXiv:2503.04490). arXiv. https://doi.org/10.48550/arXiv.2503.04490.

[redacted]

90. Modern LLMs are characterized by their scale and general-purpose architecture. They typically have trillions of parameters, and their training datasets include a wide range of sources, including books, software code, web articles, social media posts, and other content obtained from the internet. The models are often trained on text across a range of natural languages, as well as source code in a range of programming languages. The diversity of this material enables the model to encounter many different styles and topics, which expands the types of tasks that post-processing can enable. It is this scale of both data and parameters that gives LLMs the ability to respond to new prompts and generate new text across new, unseen contexts.

91. Modern LLMs are trained on corpora of extraordinary scale and diversity. The literature regarding openly available models provides insight into the scale of data required to train frontier models and confirms that training data requirements have increased dramatically (by roughly two orders of magnitude) over the course of just a few years.

92. Meta's Llama 2 model, released in July 2023, was trained on 2 trillion tokens.[52] For perspective, the Library of Congress holds about 25 million books, which is equivalent to about 2.5 trillion tokens.[53] Its successor, Llama 3, released in July of 2024, increased the number of tokens for pre-training by nearly 10-fold to 15 trillion tokens.[54] Another family of models, the Falcon models, developed by the Technology Innovation Institute in Abu Dhabi, were released in November 2023, with their flagship model requiring over 3.5 trillion tokens.[55] Google DeepMind's Gemma 7B (February 2024) was trained on 6 trillion tokens,[56] and its successor

[52] Touvron, H., et al. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models (arXiv:2307.09288). arXiv. https://doi.org/10.48550/arXiv.2307.09288.
[53] *See* General Information about the Library of Congress. (n.d.). Library of Congress. Retrieved April 15, 2026, from https://www.loc.gov/about/general-information/, Assuming ~100,000 tokens per book, 25 million X 100,000 = 2.5 trillion.
[54] Grattafiori, A., et al. (2024). The Llama 3 Herd of Models (arXiv:2407.21783). arXiv. https://doi.org/10.48550/arXiv.2407.21783.
[55] Almazrouei.E., et al. (2023). The Falcon Series of Open Language Models. (arXiv:2311.16867). arXiv. https://doi.org/10.48550/arXiv.2311.16867.
[56] Gemma Team & Google Deepmind. (2024). Gemma: Open Models Based on Gemini Research and Technology (arXiv:2403.08295). arXiv. https://doi.org/10.48550/arXiv.2403.08295.

[redacted]

Gemma 2 27B (June 2024) was trained on 13 trillion tokens.[57] Alibaba's Qwen series illustrates the trajectory most sharply: Qwen (September 2023) was trained on up to 3 trillion tokens,[58] Qwen2 (June 2024) on over 7 trillion tokens,[59] and Qwen2.5 (September 2024) on 18 trillion tokens—a six-fold increase in a single year.[60] Other frontier models from this period include Databricks' DBRX (March 2024) at 12 trillion tokens,[61] IBM's Granite 3.0 (October 2024) at 12 trillion tokens,[62] 01.AI's Yi-34B (November 2023) at 3.1 trillion tokens,[63] Baichuan 2 (September 2023) at 2.6 trillion tokens,[64] AI2's OLMo 2 13B (December 2024) at approximately 5 trillion tokens.[65] Notably, several major frontier model developers, including OpenAI, Anthropic (Claude), Google (Gemini), and Mistral AI, declined to disclose their models' pre-training token counts, suggesting that the actual frontier of data scale may be higher still than the publicly reported figures. In sum, the publicly disclosed pre-training token counts for frontier models increased from approximately 2 trillion tokens in 2023 to 18 trillion tokens by late 2024, reflecting an industry-wide technical consensus that building capable, general-purpose language models requires training data of extraordinary and ever-increasing scale.

93. The published literature also confirms that the variety of sources in these training corpora is as significant as their scale. Frontier LLMs are not trained on any

---

[57] Gemma Team & Google Deepmind. (2024). Gemma 2: Improving Open Language Models at a Practical Size (arXiv:2408.00118). arXiv. https://doi.org/10.48550/arXiv.2408.00118.
[58] Bai, J., et al. (2023). Qwen Technical Report (arXiv:2309.16609). arXiv. https://doi.org/10.48550/arXiv.2309.16609.
[59] Yang, A., et al. (2024). Qwen2 Technical Report (arXiv:2407.10671). arXiv. https://doi.org/10.48550/arXiv.2407.10671.
[60] Yang, A., et al. (2025). Qwen2.5 Technical Report (arXiv:2412.15115). arXiv. https://doi.org/10.48550/arXiv.2412.15115.
[61] The Databricks AI Research Team. (2024, March 27). Introducing DBRX: A New State-of-the-Art Open LLM . Databricks. https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm.
[62] Granite Team, IBM. (2024, October). Granite 3.0 Language Models . GitHub. https://github.com/ibm-granite/granite-3.0-language-models.
[63] Young, A., et al. (2025). Yi: Open Foundation Models by 01.AI (arXiv:2403.04652). arXiv. https://doi.org/10.48550/arXiv.2403.04652.
[64] Yang, A., et al. (2025). Baichuan 2: Open Large-scale Language Models (arXiv:2309.10305). arXiv. https://doi.org/10.48550/arXiv.2309.10305.
[65] Walsh, P., et al. (2025). 2 OLMo 2 Furious (arXiv:2501.00656). arXiv. https://doi.org/10.48550/arXiv.2501.00656.

[redacted]

single type of content; they are trained on data that spans dozens of categories, languages, and modalities. Meta's Llama 2 training corpus drew from "a new mix of data from publicly available sources," with the developers describing a deliberate strategy of "up-sampling the most factual sources in an effort to increase knowledge and dampen hallucinations."[66] Llama 3's corpus of 15 trillion tokens was processed through an elaborate pipeline involving deduplication, heuristic filtering, and model-based quality classifiers, each a distinct stage designed to improve data quality, and the developers used a language identification model to categorize documents into 176 languages.[67] Google DeepMind's Gemma 2 was trained on tokens drawn from "a variety of data sources, including web documents, code, and science articles."[68] Alibaba's Qwen2.5 increased its pretraining data from 7 trillion to 18 trillion tokens with a particular focus on improving data quality in knowledge, coding, and mathematics.[69] IBM's Granite 3.0 dense models were trained on 10 to 12 trillion tokens of language and code data sourced from different domains, with the model card identifying training data spanning 12 natural languages and multiple programming languages.[70] The BigCode consortium's StarCoder2 was trained on The Stack v2, which spans over 600 programming languages and also includes GitHub issues, pull requests, Kaggle and Jupyter notebooks, code documentation, and other natural language datasets related to math, coding, and reasoning.[71] The consistent pattern is that training corpora are assembled to be as diverse as possible across domains, languages, and formats.

---

[66] Touvron, H., et al. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models (arXiv:2307.09288). arXiv. https://doi.org/10.48550/arXiv.2307.09288.
[67] Grattafiori, A., et al. (2024). The Llama 3 Herd of Models (arXiv:2407.21783). arXiv. https://doi.org/10.48550/arXiv.2407.21783.
[68] Gemma Team & Google Deepmind. (2024). Gemma 2: Improving Open Language Models at a Practical Size (arXiv:2408.00118). arXiv. https://doi.org/10.48550/arXiv.2408.00118. At 3.
[69] Yang, A., et al. (2025). Qwen2.5 Technical Report (arXiv:2412.15115). arXiv. https://doi.org/10.48550/arXiv.2412.15115. At 4.
[70] Granite Team, IBM. (2024, October). Granite 3.0 Language Models . GitHub. https://github.com/ibm-granite/granite-3.0-language-models ; Ibm-granite/granite-3.0-8b-base . (n.d.). HuggingFace. Retrieved April 19, 2025, from https://huggingface.co/ibm-granite/granite-3.0-8b-base.
[71] Lozhkov, A., et al. (2024). StarCoder 2 and The Stack v2: The Next Generation (arXiv:2402.19173). arXiv. https://doi.org/10.48550/arXiv.2402.19173. At 2.

[redacted]

94. The volume and variety of data types and sources OpenAI used for each iteration of model pre-training reflects the vast scale and diversity of data required to train a general-purpose language model.[72] GPT-3 was pre-trained on approximately [redacted], with an underlying data mix containing approximately [redacted] of web data, books, and Wikipedia.[73]

95. GPT-3.5's pre-training data mix contained approximately [redacted] and the model was pre-trained on approximately [redacted] (epoch weighted, meaning the number of tokens used to train the dataset over all training epochs) drawn from filtered English-language web data, foreign-language web-data, Wikipedia, StackExchange, arXiv scientific papers, GitHub code repositories, and other sources spanning both natural language across multiple domains and computer source code.[74]

96. GPT-4's pre-training data mix contained approximately [redacted], and the model was pre-trained on approximately [redacted] (epoch weighted).[75] The GPT-4 corpus drew from an even broader range of sources than GPT-3.5, with web data accounting for [redacted] of epoch weighted training tokens, GitHub code repositories accounting for [redacted] and the remaining [redacted] distributed across YouTube and podcast audio-speech-recognition transcripts, Internet Archive books, Reddit and Discourse forum posts, Wikipedia in multiple languages, arXiv scientific

---

[72] Hurst Decl. Ex. 161 OpenAI's Supplemental Response to Tremblay's 1 Set of Interrogatories, 3 Supplemental Response to Interrogatory 1 at 17 ("OpenAI responds that each of GPT 3.5, GPT-3.5 Turbo, and GPT-4 was trained "from scratch," such that the use of particular training data to train, for example, GPT-3.5, does not mean that training data was also used to train, for example, GPT-3.5 Turbo.").

[73] Hurst Decl. Ex. 196, MSFT_AI_MDL_000205724; Hurst Decl. Ex. 227, Apr. 13, 2026 Declaration of Alex Paino.

[74] Hurst Decl. Ex. 227 Apr. 13, 2026 Declaration of Alex Paino; Hurst Decl. Ex. 215, OPCO_MDL_004424223, at -226. [redacted]; Hurst Decl. Ex. 111, Dec. 9, 2025, 30(b)(6) Deposition of Ian Sohl at 448:16-20; Hurst Decl. Ex. 111.A, Sohl Deposition Ex. 420; Hurst Decl. Ex. 161, OpenAI's Supplemental Response to Tremblay's 1 Set of Interrogatories, 2 Supplemental Response to Interrogatory 1 at 11-16 (tables providing training data sources used for GPT- 3.5).

[75] Hurst Decl. Ex. 227 Apr. 13, 2026 Declaration of Alex Paino; Hurst Decl. Ex. 111, Dec. 9, 2025, 30(b)(6) Deposition of Ian Sohl at 448:24-25; Hurst Decl. Ex. 111.A, Sohl Deposition Ex. 420. (Listing the training data used for GPT-3.5); Hurst Decl. Ex. 197, MSFT_AI_MDL_000221039 at -042. [redacted].

[redacted]

papers, StackExchange, PubMed biomedical literature, Quizlet, patent documents, mathematical datasets, chess game records, and numerous other specialized sources.[76] GPT-4 model-development materials reflect a shift away from trying to identify high-value source categories and toward improving overall corpus quality by removal of spam/junk materials through automated filtering, deduplication, and related heuristics.[77] GPT-4 Turbo is a fine-tuned GPT-4 and accordingly was trained using the same voluminous and diverse training data corpus as the GPT-4 model family, with additional fine-tuning datasets curated to enhance the model's multimodal capabilities and recency of knowledge.[78]

97. GPT-3.5 Turbo was likewise pre-trained using datasets orders of magnitude larger and more diverse than the training data mixes and final training datasets used for the GPT-3 and GPT 3.5 model families.[79] GPT-3.5 Turbo's pretraining

---

[76] Hurst Decl. Ex. 197, MSFT_AI_MDL_000221039 at –1042 [redacted]; Hurst Decl. Ex. 198, MSFT_AI_MDL_000221643 (DSB document; colang-v2 dataset components); Hurst Decl. Ex. 213, OPCO_MDL_001938121 ([redacted] data details spreadsheet, including tokens, weights, antiregurgitation thresholds); Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 44:24-45:9.

[77] *See* Hurst Decl. Ex. 197, MSFT_AI_MDL_000221039, at -048. [redacted] (describing spam, noise, and poor-OCR oriented exclusions); *See also* Hurst Decl. Ex. 16, Jan. 1, 2026 30(b)(1) Deposition of Greg Brockman at 77:17-78:5 [redacted]

[78] Hurst Decl. Ex. 227, Apr. 13, 2026 Declaration of Alex Paino (GPT-4 Turbo's pretraining data mix and token counts are identical to GPT-4's); Hurst Decl. Ex. 161, OpenAI's Supplemental Response to Tremblay's 1 Set of Interrogatories, 3 Supplemental Response to Interrogatory 1 at 17 (OpenAI further states that GPT-4 Turbo is a fine-tuned GPT-4 model."); Hurst Decl. Ex. 193, MSFT_AI_MDL_001742229 [redacted] with modifications including multiquery and reduced layers employing fully dense attention; resulting model then trained with [redacted] from new dataset); Hurst Decl. Ex. 220, OPCO_MDL_004661105 slides 3, 17 (GPT-4 Turbo is a [redacted] finetune of GPT-4(V); trained with [redacted] dataset containing data through April 2023); Hurst Decl. Ex. 111, Dec. 9, 2025, 30(b)(6) Deposition of Ian Sohl at 449:19-21 (GPT-4 Turbo trained using [redacted] dataset); Hurst Decl. Ex. 111.A, Sohl Deposition Ex. 420; Hurst Decl. Ex. 221, OPCO_MDL_004661108 ([redacted] data components, including datamix percentages, weights, tokens, token counts by category).

[79] Hurst Decl. Ex. 130, Nov. 18, 2025 30(b)(1) Deposition of Qiming Yuan at 105:14-15 (GPT-3.5 Turbo trained using a subset of GPT-4 training datasets).

[redacted]

data mix contained approximately [redacted] and the model was pre-trained on approximately [redacted] (epoch weighted).[80] The GPT-4o model family was trained using data from a broad range of sources similar to those used to pre-train GPT-4, including expanded multilingual web data, code and math data, speech/video-derived text, and multimodal/image sources.[81] GPT-4o's pretraining data mix contained approximately [redacted], and the model was pre-trained on approximately [redacted] (epoch weighted).[82] GPT-4o mini's pretraining data mix contained approximately [redacted], and it was pre-trained on approximately [redacted] (epoch weighted).[83] This enormous volume and breadth of training data is not incidental or a matter of convenience; it is a technical requirement for the LLM to achieve generalized capabilities. The diversity of source types, spanning web text, code, scientific literature, encyclopedias, forum discussions, speech transcripts, mathematical content, multilingual text, and more, demonstrates that the training data was assembled to capture the statistical properties of language and knowledge across the widest possible range of domains, not to target or exploit the content of any particular publisher or category of works.

98. Training on text from a limited range of sources would limit the ability of the model to generalize and to generate novel responses. To avoid this, developers aggregate material from multiple categories of documents into the training material, including long-form prose, technical writing, and informal conversation, among many others. The resulting model can then generate language that varies in tone, vocabulary, and structure.

---

[80] Hurst Decl. Ex. 227, Apr. 13, 2026 Declaration of Alex Paino (GPT-3.5 Turbo's pretraining data mix contained approximately 7 trillion tokens; pre-trained on approximately 12-13 trillion tokens (epoch weighted)).

[81] Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 74:10-12 (memes-v1 dataset used to train GPT- 4o); Hurst Decl. Ex. 111, Dec. 9, 2025, 30(b)(6) Deposition of Ian Sohl at 449:23-23 (same); Hurst Decl. Ex. 111.A, Sohl Deposition Ex. 420; Hurst Decl. Ex. 214, OPCO_MDL_000245100 (memes-v1 dataset).

[82] Hurst Decl. Ex. 227, Apr. 13, 2026 Declaration of Alex Paino.

[83] Hurst Decl. Ex. 227, Apr. 13, 2026 Declaration of Alex Paino.

[redacted]

99. The greater the diversity in the training corpus, the more robustly the model learns patterns that are characteristic of language as a whole rather than patterns specific to any particular source or category of text. For the same reason, no individual text or category of texts within a training corpus of this scale is necessary to the model's capability; the learned patterns are properties of the aggregate distribution, and the removal of any single work or collection of works from a corpus of trillions of tokens would not materially alter the statistical structure the model learns. Because the statistical patterns the model learns are distributed redundantly across millions of sources spanning trillions of tokens, no individual work or collection of works is singularly responsible for those patterns.

100. This is why a pre-trained LLM serves a fundamentally different purpose from any individual source text used during training. A book, article, webpage, or any collection of text may be read for the specific information, analysis, expression, or entertainment it provides, whereas a pre-trained LLM is developed as a general-purpose system that can be applied across a wide variety of tasks and new contexts, such as text and code generation, writing, and question answering across an enormously vast range of subject areas. The model's utility arises from the statistical structure learned across the full distribution of training data, not from preserving the content of any one text or any isolated subset of texts.

101. Preparing data on a large scale typically requires extensive preprocessing, which involves standardizing, deduplicating, and removing unwanted text so the training datasets adhere to a general form of conventions.[84] Such data cleaning and normalizing occur before "tokenization," the step where strings of text are separated into chunks called tokens.[85] The quality and uniformity of preprocessing strongly influence the stability and performance of the tokenizer, and ultimately the final language model.

[84] Murphy, K. P. (2022). Probabilistic Machine Learning: An Introduction (Third printing). The MIT Press. pp. 23–24.
[85] I discuss tokenization below in Section IV.B.4.

[redacted]

102. Data cleaning involves not only mechanical correction, but also content-based screening. Certain categories of material are either removed or down-weighted in order to limit bias, toxicity, or personally identifiable data. The filtering process may use automated classifiers capable of detecting harmful language and low-quality text, and it may also include a manual review of texts in a sample to verify the classifiers' results. Filtering criteria may also vary depending on the model's purpose and the applicable framework, but the aim is to ensure that there is an appropriate balance of language diversity and adherence to requirements. In addition, limiting identical duplication of text and sensitive content both increases the efficiency of the process as computational time is spent on informative examples rather than redundant or problematic data and reduces the likelihood of overfitting.

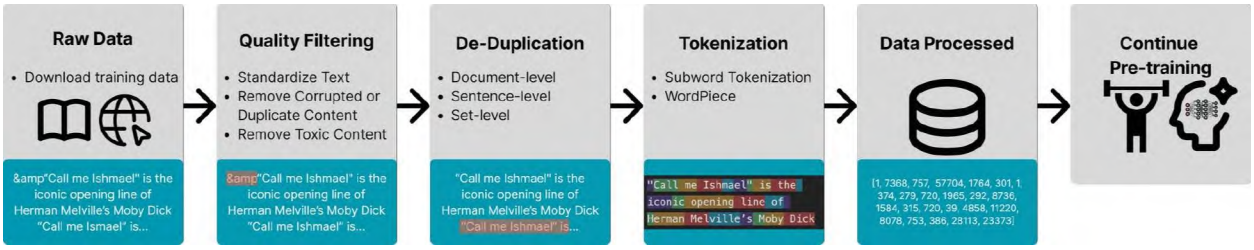
103. Once the text is cleaned and filtered, it is then compiled into standardized formats that allow it to be tokenized in preparation for model training. The data is often stored in large-scale file systems or databases that support high throughput reading for training runs. Each passage of text is accompanied by metadata, such as source category or language identifier, which assists in later analysis and quality assurance. Researchers often conduct validation studies on the filtered corpus to assess coverage, vocabulary balance, and proportion of undesired material that remains.[86] The accuracy and representativeness of the resultant corpus influence how effectively the model learns the statistical regularities of human language. Storage of data in multiple forms from the time of acquisition through completion of model training is both ordinary and necessary for the processes of research and development and model preparation.

104. I provide an example of the data preparation pipeline in Figure 9 below.

[86] Albalak, A., et al. (2024). A Survey on Data Selection for Language Models (arXiv:2402.16827). arXiv. https://doi.org/10.48550/arXiv.2402.16827.

[redacted]

Figure 9: Pre-training Data Preparation Pipeline



4. Tokenization

105. Since language models are types of neural networks that operate on numerical inputs rather than raw text, tokenization provides the bridge between ordinary language and the sequences the model processes during training and inference. This section explains what tokens are, why tokenization is used instead of a word-level vocabulary, and how raw text is converted into token sentences that can be processed by the model.

106. Tokenization is the process of taking raw, human-readable text and converting it into smaller, discrete units called tokens and then integer values.[87] Each token represents a fixed, contiguous sequence of bytes. Tokens can consist of whole words, segments of words, or single characters. A simple algorithm, described in detail below, determines which sequences of bytes correspond to tokens, based on co-occurrence statistics in a large corpus of text. As a result, individual tokens are not assigned any meaning, but rather reflect the statistical nature of how frequently characters are adjacent to each across natural language.

107. Tokenization creates a standardized and homogeneous mapping between human-written language and numerical values that serve as input to training a language model. An average 20-year-old native English-speaking adult typically knows 42,000 words,[88] roughly 12% of the over 520,000 words in the Oxford English

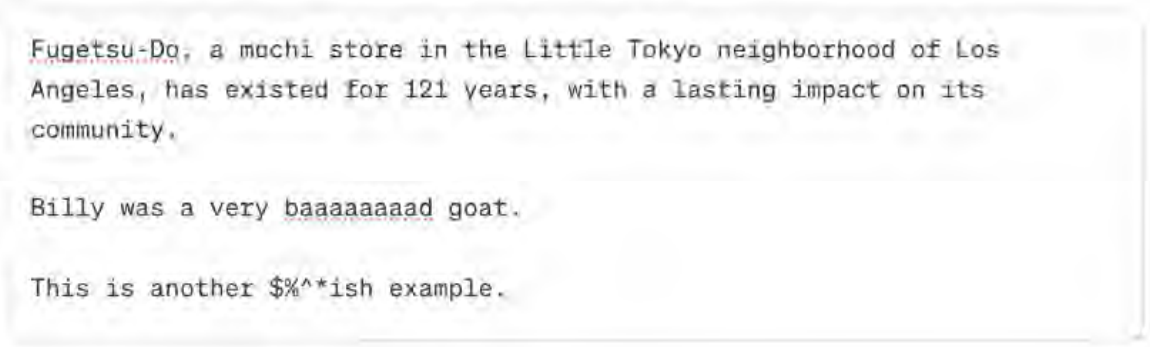
[87] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). p. 13.
[88] Brysbaert, M., et al. (2016). How Many Words Do We Know? Practical Estimates of Vocabulary Size Dependent on Word Definition, the Degree of Language Input and the Participant's Age. Frontiers in Psychology , 7 . https://doi.org/10.3389/fpsyg.2016.01116. ("[...] we estimate that an average 20-year-old native speaker of American English knows 42,000 lemmas and 4,200 non-transparent multiword expressions, [...].")

[redacted]

Dictionary.[89] A fixed vocabulary could be created and assigned to a language model. However, this vocabulary would be just as large as the number of words and not very efficient. Additionally, the model should be able to process – and generate – arbitrary text, such as misspellings and abbreviations, and be extensible. Tokenization solves these problems in modern LLMs using a process first laid out in one of the papers leading up to GPT-2.[90]

108. Tokens can be seen as the "atoms" of large language models – they are the fundamental units that an arbitrary passage of text, in any language, can be broken down into. Before describing how tokens are determined, it will be useful to describe what tokenization looks like. The figures below show output from the online tool from OpenAI.[91] When we type text, we see for example this:

Figure 10: Example of Tokenization using OpenAI's Online Tokenization Tool[92]

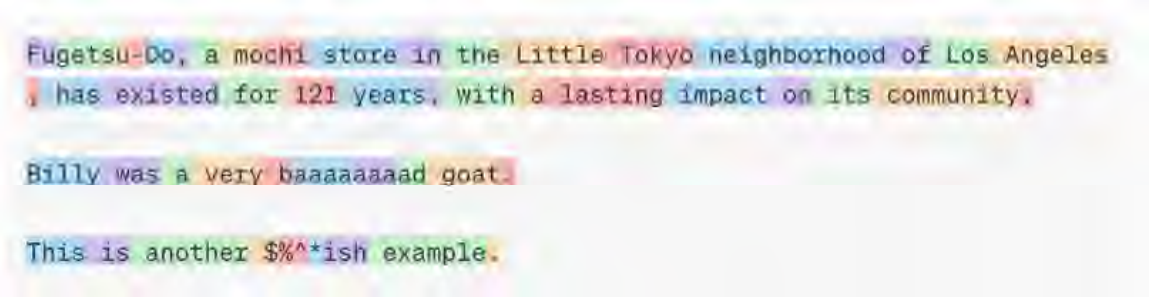


109. This text is then tokenized into a form shown here in Figure 11, where each colored sequence of characters marks a single token:

[89] Oxford English Dictionary . (n.d.). Retrieved February 27, 2026, from https://www.oed.com/search/advanced/Entries.
[90] Tokenization and the byte pair encoding method were first developed in the 2019 OpenAI GPT-2 paper, *See* Radford, A., et al (2019). Language Models are Unsupervised Multitask Learners .
[91] Tokenizer . (n.d.). OpenAI Platform. Retrieved February 24, 2026, from https://platform.openai.com/tokenizer.
[92] *Tokenizer*. (n.d.). OpenAI Platform. Retrieved February 24, 2026, from https://platform.openai.com/tokenizer.

[redacted]

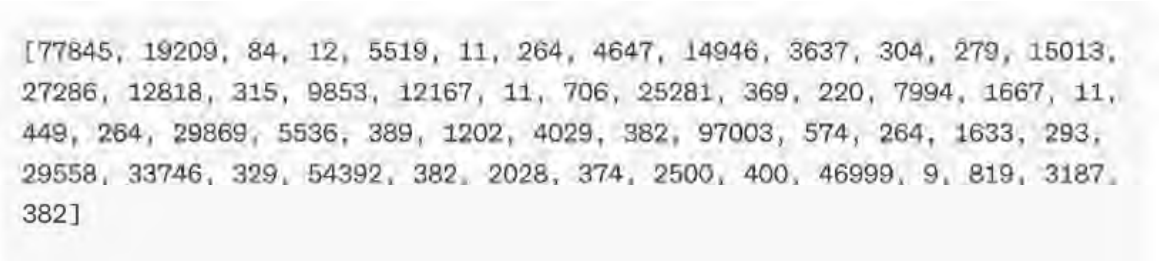
Figure 11: Tokenized Form of Input Using OpenAI's Online Tokenization Tool[93]



110. Several common words are apparent, but they often appear with a space (␣) prepended: ␣ has, ␣ for, ␣ community, etc. Other words are broken up into pieces; for example, the word ␣ mochi (with leading space) is broken up as two tokens ␣ mo + chi. Also, the string ␣ baaaaaaaad, which the algorithm may have never seen before, is broken up as the four tokens ␣ b + aaaa + aaa + ad.

111. While this shows how words are broken up into tokens, the tokens are then mapped to unique integers, and this sequence of numbers is then processed by the language model and transformed into embeddings, which I describe further below. In this example, Figure 12 shows the sequence of numbers the language model is given:

Figure 12: Token IDs of Input Using OpenAI's Online Tokenization Tool[94]



112. The tokenization process has the very important property that it is statistical and local, as the determination of the token vocabulary is based solely on the co-

[93] *Tokenizer*. (n.d.). OpenAI Platform. Retrieved February 24, 2026, from https://platform.openai.com/tokenizer.
[94] *Tokenizer*. (n.d.). OpenAI Platform. Retrieved February 24, 2026, from https://platform.openai.com/tokenizer.

[redacted]

occurrence statistics of which characters appear next to other characters in a large passage of training text. This means that tokens themselves do not explicitly carry meaning – they can only be interpreted in their relation to other tokens as occurring in the training text.

113. After the token vocabulary has been built, the tokenization pipeline runs through each document in the training data, mapping the text to a sequence of integer tokens. This is the form of the text that the LLM is trained on; the original human-readable forms of the documents are never directly used.

114. It bears repeating that because tokens are automatically constructed based solely on adjacent (side-by-side) co-occurrence statistics in a large corpus, the tokens themselves do not carry any meaning. Subsequent training of the language model implicitly ascribes information to tokens based on the broader context in which they appear.

5. Embeddings

115. Once the text has been tokenized and converted into token identifiers, which are integers, the token IDs are mapped to continuous vector representations called embeddings. This is a necessary step since the neural network models cannot readily process integers; instead, the networks need numerical values (floating point numbers) in order to apply the basic operations of linear transformation, summing, and thresholding. This section explains what embeddings are, how they represent relationships among tokens in a high-dimensional space, and how different embedding types provide the starting representations the transformer uses during training and inference.

116. A vector is the mathematical term for a list of numbers. The numbers can be seen as coordinates on a map. If we are standing in a room, the numbers (31, 42, 53) (in units of inches) can serve to specify a precise location in the room – 30 inches from a specific corner of the room along one wall, 40 inches from the corner along the other wall, and 50 inches up from the floor. A similar vector, say (33, 37, 51), represents a nearby location. To say that this other vector is close means

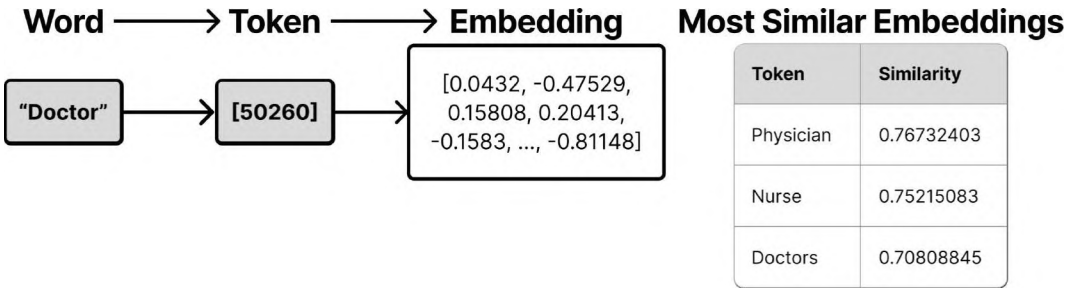
[redacted]

that the distance between them is small (in this case 5.74 inches). Closeness can also be expressed in terms of the angle formed, at the corner of the room, between two lines drawn from the corner to each of those two locations (in this case the angle is approximately 3.72 degrees). Embeddings are coordinates assigned to tokens, in a very high dimensional space. While we cannot directly visualize points in that space (as our brains only evolved to see in three dimensions), the same math applies.

117. An embedding maps a token ID to a point in a high-dimensional space where the distances and directions of vectors in the space reflect the patterns seen in the data.[95] Embeddings are learned as part of language model training.[96]

118. During training, the model learns to cluster the embeddings of tokens occurring in similar context in proximity to each other. Likewise, embeddings of tokens having quite different meanings will tend to occupy separate areas of the high-dimensional embedding space. When trained on large amounts of text, even simple bigram language models result in embeddings that exhibit rich and interesting relationships, both semantic and syntactic, which I illustrate in Figure 13 below.

Figure 13: Example of Embedding Process and Similar Embeddings



119. The word "Doctor" is transformed into its tokenized integer representation, which is then embedded resulting in a list of numbers. Represented as points in a high-dimensional space, the words with similar embedding vectors, as measured

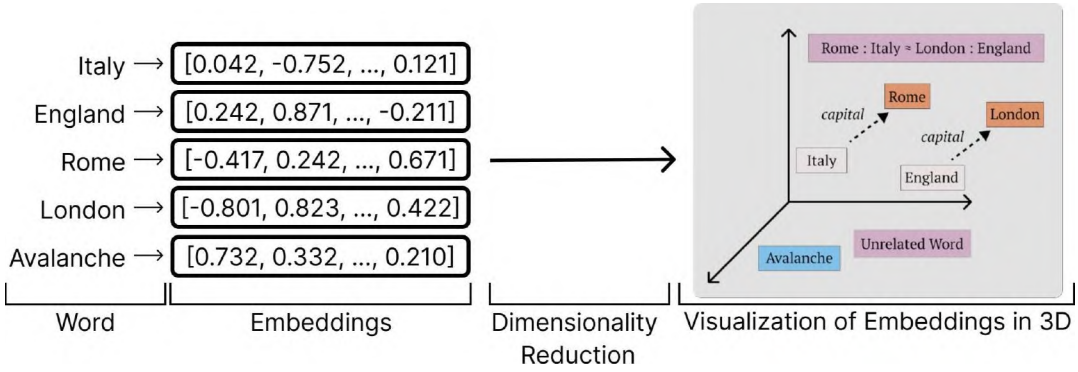
[95] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). pp. 99–100, 111–113.
[96] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). p. 96.

[redacted]

by the cosine of the angle between them, have both syntactic (plural or singular nouns) and semantic information (related as medical providers).

120. Figure 14 below provides an example illustration of how relationships are embedded in 3D space. Related words, such as "Italy" and "England," create vectors that are a similar distance apart in space, while unrelated words, such as "Avalanche", would lie further away relative to the overall embedding space. One can imagine the equal distance between "Italy" and "England" and "Rome" and "London" representing an embedding vector in which capitals of countries are located. The angles of these vectors all depend on not only each other, but rather are distributed across all dimensions in the embedding vectors.

Figure 14: Example of Embeddings in Space



121. It is important to recall that token IDs are based only on adjacent co-occurrence statistics, using a procedure that recursively merges tokens that frequently appear side-by-side. Starting from the token IDs, the embedding vectors are randomly initialized and then updated to better predict nearby words by training the language model. Similar to tokenization, the "meaning" the embeddings carry results solely from this prediction task. ␣ ␣ For instance, the vectors for the tokens " doctor" and " nurse" might shift to be closer together since they appear in contextual relationships to each other, while the vector for ␣ " bird" remains further away. Any interpretability of the token embeddings is solely an artifact – a side effect of the training process to maximize predictability. Any appearance that the resulting vectors "understand" meaning is a statistical artifact: a side effect of optimizing for predictability across a vast and diverse corpus of text.

[redacted]

122. Further, embeddings are not limited to individual tokens. Modern architectures such as transformers incorporate other types of embeddings that can each capture different linguistic or structural aspects of the input.[97] Token (or word) embeddings represent a word's semantic meaning as a dense vector, as discussed above. A different type of embedding called positional embedding encodes the position of tokens in a sequence.[98] This allows the model to identify and distinguish identical words in the same input that appear in different positions.

123. These embeddings serve as the foundation for the entire model. During both training and inference, the model obtains the associated vector of a given input token from the embedding layer, which serves as the initial representation that the network processes.[99] Successive layers then transform these vectors into increasingly abstract and contextual features.[100]

6. Transformer Architecture and Attention

124. The central challenge in modeling human language is to identify, represent, and use the important relationships between words in context, even when those words are far apart. The transformer was developed to improve how neural networks capture and represent such relationships.[101] The transformer is used in nearly all modern LLMs and is a neural network for natural language that processes all tokens in an input simultaneously, computing the relationships among every pair of tokens in the sequence at each layer, allowing it to capture context, long-distance relationships, and subtle differences in meaning. This section first explains how transformers represent contextual relationships among tokens via attention. It then describes how transformer layers combine attention

[97] Devlin, J., et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (arXiv:1810.04805). arXiv. https://doi.org/10.48550/arXiv.1810.04805.
[98] Murphy, K. P. (2022). Probabilistic Machine Learning: An Introduction (Third printing). The MIT Press. p. 526.
[99] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). pp. 135–136, 172.
[100] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). pp. 171–172.
[101] Vaswani, A., et al. (2023). Attention Is All You Need (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762.

[redacted]

and feed-forward processing during training and inference to support next-token prediction.

125. The transformer is composed of a stack of layers. Each layer computes a new, "transformed" embedding of each input token, which encodes different aspects of the context of that token. The transformer starts from the context-independent embeddings of each token before tailoring the meaning to the context in deeper transformer layers. Thus, the embedding for a token like "leaves," which could refer to either leaves from trees or the act of removing oneself, would be transformed to an embedding that represents only one of its two possible meanings depending on the context provided by the other tokens.

126. Each layer in the transformer is organized into "blocks" that implement fundamental operations in a particular order. The first operation, as described above, is known as self-attention. Self-attention evaluates the relationships among tokens in the input, and updates those embeddings so that each token's representation reflects its context.[102]

127. Self-attention computes relationships between a token and every other token, effectively creating a type of influence map across the sequence. Each token is represented by a numeric embedding vector that reflects its current state. Each embedding vector is a list of floating-point numbers of a fixed length or dimension (typically 5,000 to 20,000). The vectors are compared to compute attention weights, which indicate how strongly one should influence another as the model updates its internal representation.[103]

128. Consider the following sentence, for the purpose of illustration: "The leaves, as the Vermont weather becomes colder and the days grow shorter, turn bright orange and fall silently from the trees in October."

[102] Russell, S. J., et al. (2022). Artificial Intelligence: A Modern Approach (Fourth edition, global edition). Pearson.

[103] Vaswani, A., et al. (2023). Attention Is All You Need (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762.

[redacted]

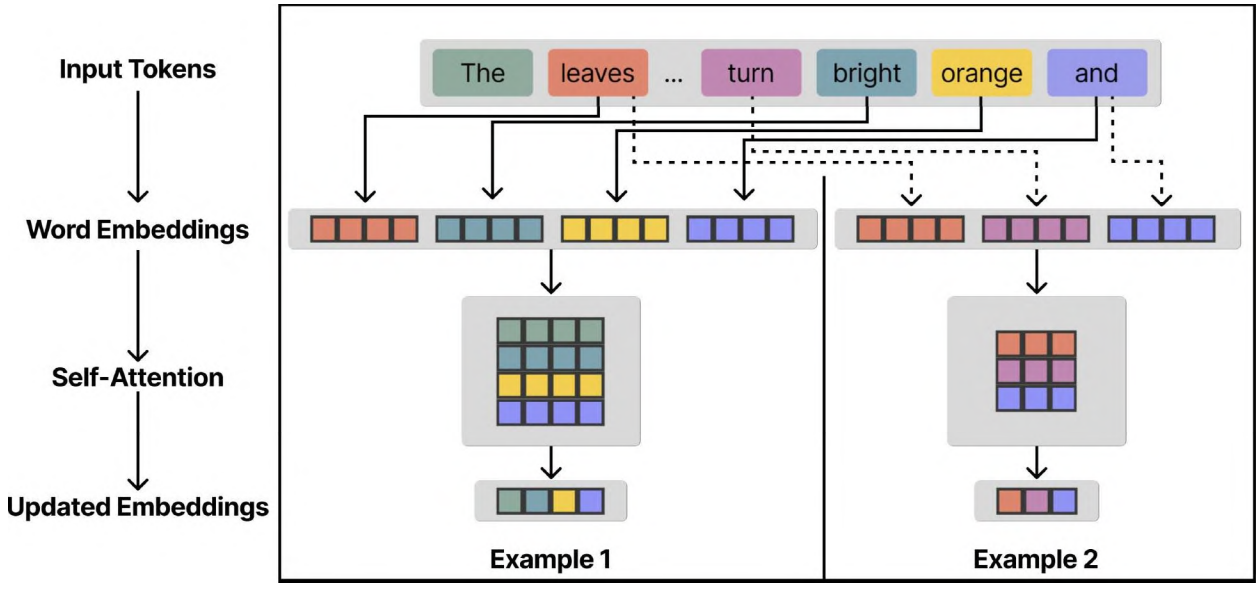
129. This sentence, like most sentences, exhibits many connections between the words. The verb "turn" has subject "leaves", and so must be plural in number (that is, "leaves turn" but "a leaf turns"). While the verb "talk" would also satisfy this constraint, leaves don't usually speak, so this verb would be unlikely. Similarly, the word "trees" is related to "leaves", and the month "October" is part of autumn.

130. A language model is tasked with predicting the next token in the sequence; for example, it needs to "fill in the blank" here: "The leaves, as the Vermont weather becomes colder and the days grow shorter, turn bright orange and ____"

131. If we focus on, or attend to, the word "orange", likely completions of the sentence might be words like "yellow" or "red". On the other hand, if we attend to the words "leaves" and "turn", likely completions might be verbs like "fall" or "drop".

132. Continuing the example above, in order to fill in the blank, the embedding vector "and" is transformed to the compound embedding vector "and" + "leaves" + "turn" + "orange", which increases the chances of words like "yellow", "red", "fall", and "drop".

Figure 15: Example of Self-attention



133. In Figure 15 above, I demonstrate how self-attention identifies words (tokens) in the recent context that are predictive of the token to follow. Different attention

[redacted]

"heads", or independent instances of the self-attention mechanism, capture different types of relationships. In Figure 15's Example 1, the word following "and" is likely to be a color adjective like "red", paired with "bright" and "orange". In Example 2, the word following "and" would more likely be a verb paired with "leaves" and "turn". Both of these attention weights are used to transform the context-independent embedding "and" into a rich and specialized embedding for this context, increasing the likelihood of appropriate completions.

134. In an extension called multi-head attention, this process is repeated in multiple subspaces to enable the model to evaluate different types of relationships simultaneously, such as syntactic dependencies, semantic roles, and discourse relations.[104] In our working example above in Figure 15, two such multi-head attention relationships are illustrated.

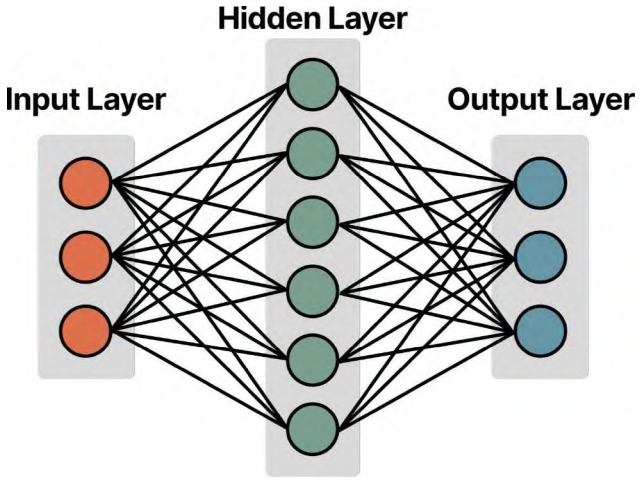
135. The second component, or block, of a transformer layer is a feed-forward neural network, also called a multi-layer perceptron ("MLP"), which I provide a visualization of in Figure 16. The weights of this neural network are randomly initialized, and then adjusted differently for each layer during the training process. The input to the neural network in a given layer is the embedding vector for a token at that layer, after the embedding vector has been transformed by self-attention.[105] The MLP applies a nonlinear transformation to process these embedding vectors before passing them to the next layer.[106] Intuitively, the purpose of the MLP is to learn new "features" that can be used by the next transformer layer, to build up increasingly rich and predictive contextual information.[107] This enables the model to generalize across a greater range of contexts including those not seen in its training data, increasing testing performance.

---

[104] Vaswani, A., et al. (2023). Attention Is All You Need (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762.
[105] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). p. 173.
[106] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). pp. 121, 179, 343.
[107] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). p. 235.

[redacted]

Figure 16: Feed-forward Neural Network

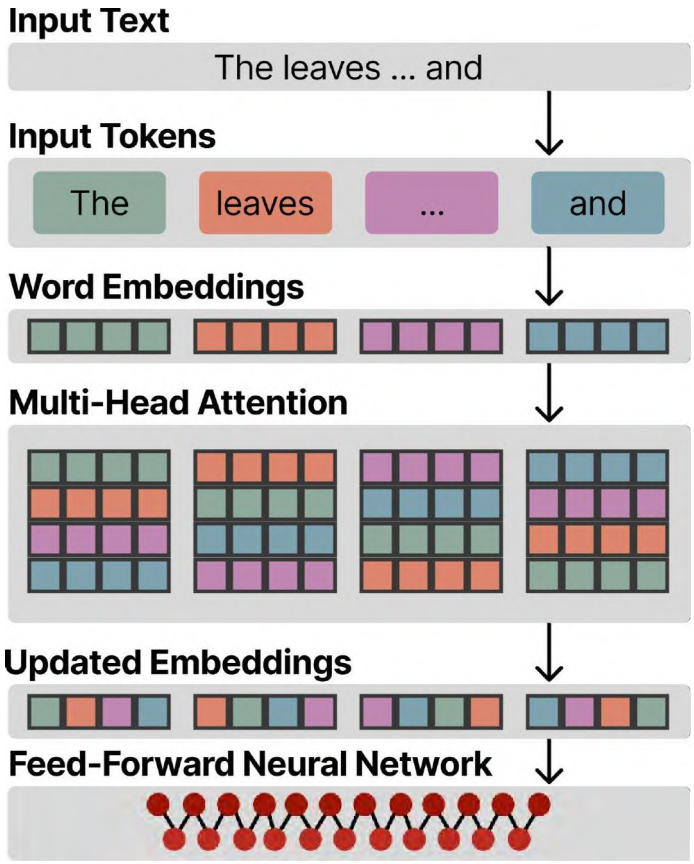


136. Using the above example, the neurons representing "foliage" and "autumn" may be taken in their conjunction while discarding less predictive components of the embeddings. The feed-forward process and the attention mechanism are the key operations that form the repeating computational pattern defining the transformer architecture.[108] In Figure 17 below, I provide a visualization of the transformer architecture.

[108] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). p. 173.

[redacted]

Figure 17: Overview of Transformer Architecture



137. During pre-training, the parameters of the language model are adjusted to optimize the prediction of the next token. The model takes in text, represented as a sequence of token IDs, and is tasked with predicting which token ID comes next in the sequence.[109] As described above, layers of attention and feed-forward neural networks inside each transformer layer learn how tokens relate to one another, by comparing their embedding vectors and transforming those vectors into a new form, which represents the important contextual information.

138. The final layer of the transformer produces a probability distribution over the entire token vocabulary.[110] If the actual next token has a high probability, the

[109] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). pp. 147-149.
[110] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). p. 155.

[redacted]

model is predicting well. On the other hand, if the next token has a low probability, the parameters will be adjusted so that this probability increases.

139. This prediction task provides the model with its learning signal. When the predicted token does not match the actual one in the training data, the training algorithm traces how each parameter contributed to the mistake through an algorithm called " backpropagation." Backpropagation adjusts those parameters to bring future predictions closer to the correct outcome.[111] This process happens continually during training as the model encounters new batches of text from the training data. Over time, billions of small adjustments shape the model's weights, which reflect the statistical relationships underlying natural language learned across the entire distribution of training text. In general, using model parameters to "memorize" long sequences of tokens in the training data will not help the model to predict well in the novel contexts that arise in test data.

140. At inference time, when the trained model generates text in response to a user's prompt, the model uses the same transformer architecture to compute a probability distribution over the next token. Given a prompt, it calculates the probabilities for the next token, chooses one, adds it to the sequence, and continues step by step. Each new prediction depends on all of the prior tokens, and coherence emerges from the accumulation of many small, locally informed decisions.[112] The quality of the generated output depends both on how accurately the model's parameters reflect the patterns it encountered across the entirety of the training data set, and on how the sampling strategy balances predictability with creative variation, which I discuss further in Section IV.B.6). Because each token is sampled from a probability distribution rather than retrieved from stored text, the same prompt can yield different outputs on different occasions.

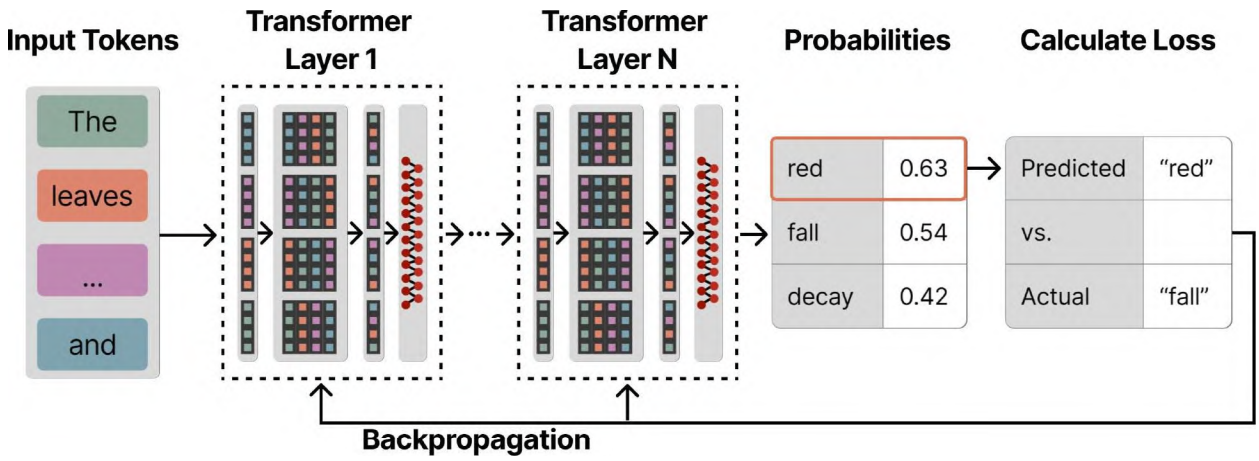
141. Predicting the next token is the only function used by the model to learn how to generate realistic looking language. When trained at scale on large volumes of text, the model will learn to generate fluent natural language that is essentially

[111] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). p. 139.
[112] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). pp. 154-156.

[redacted]

indistinguishable from actual human text, not by copying or retrieving text from the training data, but by applying the statistical patterns of language encoded in the model's parameters to construct new sequences, token by token. In Figure 18 I provide an example of the next token prediction process for a pre-trained LLM.

Figure 18: Example of Token Prediction Process



142. Transformer-based LLMs can differ not only in size, but also in how they allocate computation within the model. In a traditional dense design, the same feed-forward parameters at a given layer are applied to every token representation. In a mixture-of-experts ("MoE") design, by contrast, certain feed-forward layers are replaced with a collection of specialized subnetworks, or "experts," along with a routing mechanism that selects only a small subset of those experts for a given input.[113]

143. A dense model, such as those I've described above, must learn broadly useful statistical structure in a single shared set of parameters, which in turn calls for training on a large corpus of text. In an MoE model, different experts are trained on different routed inputs and can develop different forms of specialization, so broad and sufficiently varied training data can help enable useful expert specialization.[114] Additionally, some designs can increase model capacity without

[113] Bergmann, D. (n.d.). What is mixture of experts? IBM. Retrieved April 9, 2026, from https://www.ibm.com/think/topics/mixture-of-experts.
[114] Sanseviero, O., et al. (2023, December 11). Mixture of Experts Explained. HuggingFace. https://huggingface.co/blog/moe.

[redacted]

a proportional increase in computation or model parameters (which I describe in Section IV.B.7) below). Sparse MoE architectures, for example, have been applied in language modeling and machine translation to increase model capacity without overfitting.[115]

7. Model Parameters

144. Parameters, as introduced in Section IV.A.4, are the adjustable numbers that determine how a model processes input and produces output. This section briefly describes what those parameters are, why models contain so many of them, and why that scale affects the computational resources required for training.

145. The most important and common parameters in a neural network are weights, which influence how information flows from one layer to the next. During training, these weights are adjusted to enable the model to better predict the next token across a wide range of contexts. At the conclusion of training, these weights reflect the statistical relationships underlying natural language learned across the entire distribution of training text.

146. Modern language models can contain trillions of such weights, and this scale gives them the capacity to capture subtle and intricate patterns across an enormous range of different types of text.

147. Because the network is arranged in interconnected layers, as described in Section IV.B.6, scaling the number of these weights and layers can dramatically increase the complexity of the system. This increased complexity leads to an increase in the resources required to train the model.

8. Scaling Laws

148. Scaling laws describe the observed relationship between a model's performance and the resources used to train it.[116] In the context of LLMs, scaling laws provide

[115] Shazeer, N., et al. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer (arXiv:1701.06538). arXiv. https://doi.org/10.48550/arXiv.1701.06538.
[116] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). pp. 190–191.

[redacted]

an empirical framework for understanding how performance changes as model size, dataset size, and computational power increase together. This section explains the basic logic of scaling laws, and the way scaling behavior is evaluated in practice.

149. Researchers found that when model size, dataset size, and computational power grow in roughly equal measure, performance improves in a predictable way. The drop in training loss tends to follow a power-law curve, where each order of magnitude increase in scale brings a constant improvement in loss, which is the predictive accuracy of the model.[117] These findings have helped guide the design of large language models, allowing developers to estimate the most effective model size for a given amount of data and available compute.

150. The central principle of scaling laws is proportional growth. When a model's parameters are increased without a corresponding expansion in data or computational resources, performance gains taper off as the network begins to overfit or fails to converge. Conversely, enlarging the dataset without increasing model capacity can prevent the system from fully leveraging the available information. Optimal scaling therefore requires that model size, data volume, and training duration grow together in proportion. Balance among these factors produces a predictable performance curve that allows model creators to estimate model outcomes before training begins, significantly saving resources and reducing the expense of searching for optimal models.

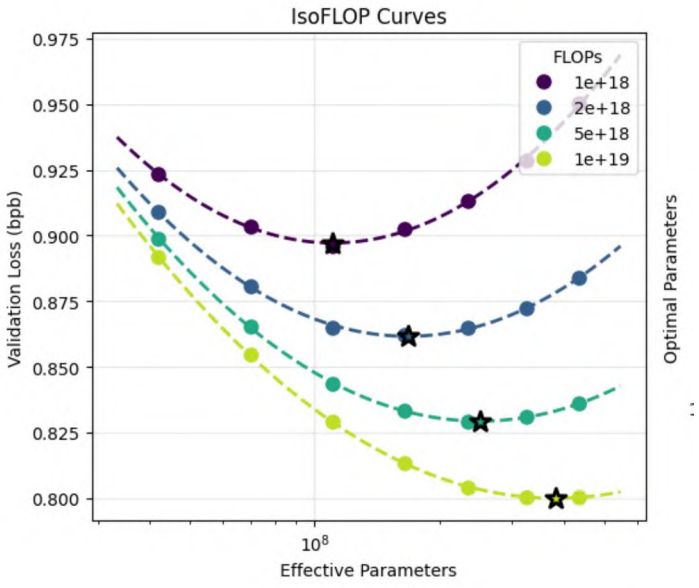
151. Figure 19 below shows the scaling laws observed for LLMs that can be trained cheaply with limited resources.[118] The laws observed at industrial scale are qualitatively and quantitatively similar.

[117] Kaplan, J., et al. (2020). Scaling Laws for Neural Language Models (arXiv:2001.08361). arXiv. https://doi.org/10.48550/arXiv.2001.08361.

[118] The scaling laws observed at industrial scale are qualitatively the same, but with different parameters.

[redacted]

Figure 19: Examples of Scaling Laws[119]



152. A scaling law relates the predictive performance of a language model to the computational resources required to train that model. Performance is evaluated in terms of how well the model predicts previously unseen validation data. Using data different from the training data ensures that the model can be evaluated for underfitting or overfitting. In particular, using model parameters to memorize training data "verbatim" will generally not be advantageous for predicting tokens in the new test data.

153. In the leftmost plot above, the x-axis indicates the number of parameters in the model, and the y-axis shows the predictive performance in terms of "bits-per-byte" ("bpb"). This is an information-theoretic metric used to evaluate and compare the efficiency of language models.

9. Generalization

154. Generalization refers to a model's capacity to apply what it has learned from training data to new, unseen inputs. For LLMs, this ability to generalize depends

[119] Karpathy, A. (2025). *nanochat: The best ChatGPT that $100 can buy*. GitHub. https://github.com/karpathy/nanochat.

[redacted]

on how the network's parameters capture relationships and pattern from the data rather than "memorizing" long passages of the training text. This section explains how generalization arises during training, how it differs from memorization, and how it is evaluated using held-out data and benchmark tests.

155. During training, the model encounters a wide range of phrases, sentences, subjects, topics, and styles. When generalization is strong, the model can generate coherent and contextually appropriate text about unfamiliar subjects because its parameters represent the deeper structure of language rather than the surface content of the dataset. For LLM outputs, the model generates new token sequences from probability distributions adapted to the local context, producing novel text that is distinct from the source material used for training.

156. During training, LLMs do not learn or retain information on a per-passage basis. Instead, they update their parameters incrementally in response to associations and regularities observed across many texts and contexts. For example, an LLM may be trained on multiple passages related to "Sacramento." As the model repeatedly encounters this word near other words like "California," "capital," "river," or "Northern," its parameters are adjusted to reflect relationships that hold across many examples rather than features unique to any one passage about Sacramento. The same principle applies more broadly. If the model is trained on many passages about "hurricanes," it may learn that words like "storm," "wind," "landfall," and "evacuation" are often occurring in related contexts. If it is trained on many passages about "contracts," it may learn recurring relationships among terms such as "party," "agreement," "breach," and "damages." Capturing these recurring associations reduces training loss across many examples, whereas memorizing a single passage generally does not. As a result, the model can use those learned relationships in new or previously unseen contexts, which reflects generalization rather than memorization of training text.

157. Instances of memorization are more likely to occur under particular technical conditions that place unusual weight on specific examples rather than broader statistical structure. One such condition is highly duplicated data, where the

[redacted]

same or nearly identical passages appear many times in the training corpus, which increases the incentive for the model to assign disproportionate importance to those sequences.[120] Memorization may also be more likely where training examples contain unusually identifying tokens or token patterns, such as unique identifiers, numerals, or proper nouns, which can make particular examples easier for the model to distinguish and later reproduce.[121] In addition, adversarial prompting and extraction-style attacks can be designed to increase the likelihood of atypical outputs, including outputs that resemble specific training examples. These conditions are relevant to the study of memorization, but they do not alter the key point that the dominant objective of training is to capture generalizable patterns. And these conditions share a common characteristic. They are atypical of ordinary model use.

158. Since LLMs capture relationships in their training data, they may produce facts or concepts that also appear in that data. In some cases, a model may generate a sequence of text that exists verbatim in the training material. However, such behavior does not by itself imply that the model stores specific training passages. The distinction between a properly trained model and one that overfits its training data may not always be clear from the model outputs alone, particularly where repeated training examples, outlier pieces of text, or adversarial prompts increase the likelihood of reproduction. This type of replication of training sequences could also take place with the much simpler n -gram type of language models used in the early 2000s; it is not due to the neural network architecture.[122] In practice, modern LLMs employ various decoding methods, such as sampling-based generation, to reduce repetitive, deterministic outputs and encourage variation in responses.[123] This allows LLMs to express the same

[120] Kandpal, N., et al. (2022). Deduplicating Training Data Mitigates Privacy Risks in Language Models (arXiv:2202.06539). arXiv. https://doi.org/10.48550/arXiv.2202.06539.
[121] Tirumala, K., et al. (2022). Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models (arXiv:2205.10770). arXiv. https://doi.org/10.48550/arXiv.2205.10770.
[122] Brants, T., et al. (2007). Large Language Models in Machine Translation . https://aclanthology.org/anthology-files/anthology-files/pdf/D/D07/D07-1090.pdf.
[123] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). pp. 154–157.

[redacted]

underlying information in distinct ways. For example, an LLM trained on material from a math book may answer the prompt "What is 101+202?" with "303" as the response because the output best reflects the model's learned representation of arithmetic rules, rather than copying a specific question and answer pair from the training material.

159. Of course, simple arithmetic questions may also appear in large training datasets, so it may be unclear from output alone whether the model generalized or memorized the response. The distinction can be illustrated more clearly with a novel rule supplied in the user's prompt itself, which tests abstract pattern induction of LLMs.[124] For example, suppose a model is instructed to transform words by reversing their letters, moving the last two letters to the front, and then appending the suffix "ay," and is given examples such as:

a. "lamp" → "alpmay"

b. "bird" → "ibdray"

c. "horizon" → "ohnoziray"

d. "language" → ?

160. If the model then transforms "language" into "alegaugnay," that output reflects the model's application of an abstract pattern presented in context to a previously unseen input, rather than memorization of a stored training passage.

161. A model's ability to generalize can be evaluated using held-out datasets that contain material unseen during training.[125] Performance on these sets indicates the extent to which a model's internal representations extend to new data. A model that achieves comparable results on both the training and testing corpora is considered to generalize effectively, whereas one that performs well only on familiar material is regarded as overfit. Generalization is also assessed through

[124] Webb, T., et al. (2023). Emergent Analogical Reasoning in Large Language Models (arXiv:2212.09196). arXiv. https://doi.org/10.48550/arXiv.2212.09196.

[125] *See* Section IV.A.5; See also, Goodfellow, I., et al. (2016). Deep Learning. The MIT Press. pp. 110, 114–115.

[redacted]

targeted benchmarks that test reasoning, factual accuracy, and cross-domain transfer, which provide a broader view of how knowledge acquired during training supports performance on new tasks.[126]

Figure 20: LLM Evaluation Results[127]

Benchmark (shots)	GPT-3.5	GPT-4	PaLM	PaLM-2-L	LLAMA 2
MMLU (5-shot)	70.0	**86.4**	69.3	78.3	68.9
TriviaQA (1-shot)	–	–	81.4	**86.1**	85.0
Natural Questions (1-shot)	–	–	29.3	**37.5**	33.0
GSM8K (8-shot)	57.1	**92.0**	56.5	80.7	56.8
HumanEval (0-shot)	48.1	**67.0**	26.2	–	29.9
BIG-Bench Hard (3-shot)	–	–	52.3	**65.7**	51.2

162. The table illustrated in Figure 20 demonstrates evaluation results for several LLMs, with their performance compared across a set of standardized benchmark tests. The benchmarks assess different capabilities, such as broad subject-matter knowledge (MMLU),[128] factual question answering (TriviaQA[129] and Natural Questions[130]), mathematical reasoning (GSM8K),[131] code generation (HumanEval),[132] and reasoning tasks drawn from diverse domains (BIG-Bench Hard).[133] The reported scores indicate the percentage of benchmark questions answered correctly be each model. In this context, the term "shots" refers to the number of example questions provided to the model as part of the prompt, which is a standard evaluation technique used to assess a model's ability to generalize

[126] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). pp. 163–166.
[127] Touvron, H., et al. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models* (arXiv:2307.09288). arXiv. https://doi.org/10.48550/arXiv.2307.09288.
[128] Hendrycks, D., et al. (2021). Measuring Massive Multitask Language Understanding (arXiv:2009.03300). arXiv. https://doi.org/10.48550/arXiv.2009.03300.
[129] Joshi, M., et al. (2017). TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension (arXiv:1705.03551). arXiv. https://doi.org/10.48550/arXiv.1705.03551.
[130] Kwiatkowski, T., et al. (2019). Natural Questions: A Benchmark for Question Answering Research. Transactions of the Association for Computational Linguistics , 7 , 453–466. https://doi.org/10.1162/tacl_a_00276.
[131] Cobbe, K., et al. (2021). Training Verifiers to Solve Math Word Problems (arXiv:2110.14168). arXiv. https://doi.org/10.48550/arXiv.2110.14168.
[132] Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code (arXiv:2107.03374). arXiv. https://doi.org/10.48550/arXiv.2107.03374.
[133] Srivastava, A., et al. (2023). Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models (arXiv:2206.04615). arXiv. https://doi.org/10.48550/arXiv.2206.04615.

[redacted]

from limited guidance on the task. The breadth of these benchmarks reflects the diversity of capability of LLMs, which is the direct product of generalization--parameters that have captured the statistical structure of language broadly enough to support performance across tasks that bear no resemblance to one another, and none of which involving reproducing the text on which the model was trained.

10. Emerging Literature on Memorization

163. Recent empirical work formalizes the distinction between memorization and generalization in LLMs, and emphasizes that reproducing training-like text does not, by itself, establish memorization.[134] In that literature, memorization is often separated into sample-specific retention and broader generalization about the underlying data-generating process. This section explains that distinction, why generalization is understood to dominate model behavior at scale, and the limits of inferences that can be drawn from extraction-based studies alone. The studies discussed below address LLM behavior as a general phenomenon. They examine a range of model architectures and scales, but their findings are directly relevant to the models at issue in this case, which use the same transformer architecture, the same self-supervised training objective, and training corpora of comparable or greater scale.

164. The literature surrounding memorization identifies technical circumstances under which memorization is more likely to occur. These include repeated exposure to the same or nearly identical training passages, as well as training examples that contain unusually identifying tokens or token patterns that make particular examples easier to distinguish and later reproduce. Researchers also study memorization through adversarial prompting and extraction attacks, which are designed to induce outputs resembling training text that would not ordinarily be produced in typical use. In that sense, memorization is a real but bounded

[134] Morris, J. X., et al. (2025). How much do language models memorize? (arXiv:2505.24832). arXiv. https://doi.org/10.48550/arXiv.2505.24832.

[redacted]

phenomenon that depends on both the composition of the training data and the manner in which the model is probed.

165. While LLMs are capable of memorizing limited amounts of information, such memorization is constrained by the model's capacity and does not increase indefinitely as training data grows. As training datasets become very large relative to model size, models increasingly allocate their representational capacity to shared statistical structure across many examples rather than to information tied to individual passages, meaning it learns abstractions and patterns that apply broadly rather than retaining information tied to individual passages. Therefore, improvements in model performance are driven primarily by generalization to the underlying data-generating process rather than by storage of specific training instances.[135] When models are trained on sufficiently large and diverse corpora, the ability to reproduce text similar to training material reflects learned linguistic patterns rather than retention of particular examples.

166. Although studies examine the extent to which models reproduce training data through methods such as extraction attacks, some researchers argue that when such attacks succeed, this is insufficient to show that model has sample-specific retention of a particular training passage, since the text might instead be constructed by the application of broader statistical patterns that the model learned across many training examples.[136]

11. Other Optional Pre-training Steps

167. After broad pre-training and before later post-training stages, additional steps can be taken to continue model pre-training. These steps can consist of continuing to train on a more targeted distribution of data selected to strengthen specific capabilities, domains, or modes of reasoning. The datasets selected may contain domain-specific corpora, linguistic-specific corpora, instruction-

[135] Morris, J. X., et al. (2025). How much do language models memorize? (arXiv:2505.24832). arXiv. https://doi.org/10.48550/arXiv.2505.24832.
[136] Haviv, A., et al. (2026). We Should Separate Memorization from Copyright (arXiv:2602.08632). arXiv. https://doi.org/10.48550/arXiv.2602.08632.

[redacted]

formatted material, or other sources chosen to move the model toward the kinds of tasks it will be later asked to perform.

168. These additional materials are used to give stronger performance in selected domains or reasoning modes, and not to preserve, restate, or serve the original purpose of any individual source. The model's capabilities that result are not tied to the informational or expressive purpose of any individual item used during this stage.

C. LLM Post-Training and Output Generation

169. The pre-training process provides an LLM with a general understanding of statistical patterns underlying natural language, and supplementary training may further strengthen particular capabilities or domains, but additional steps are required before it can perform reliably in real-world applications. While the model on its own can produce coherent and fluent text, it can lack context, precision, factual grounding, and awareness of human expectations. Post-training methods align the pre-trained system by making smaller, targeted adjustments in the model's parameters that tailor its responses toward defined goals such as engaging in a back-and-forth chat. These methods can involve retraining the model on curated datasets, exposing it to human-rated examples, or connecting it to external sources of information. Each post-training stage involves distinct engineering decisions, data, and design choices that contribute to the functionality and value of the resulting model.

1. Fine-tuning

170. Fine-tuning is the process of continuing the training of a pre-trained model on a smaller, specialized dataset so that its outputs better reflect the goals of a specific application. Unlike pre-training, which develops broad statistical representations of language, fine-tuning adapts those representations to a narrower set of tasks or domains. This section explains how fine-tuning works, how fine-tuning data differs from pre-training data, and why that process allows a single base model to be adapted for different uses.

[redacted]

171. A pre-trained model captures the structure of language and implicitly encodes a great deal of factual information. Fine-tuning shapes the model's parameters so that the model can better respond using certain forms of expression or reasoning.[137] The goal is not to overwrite what the model has already learned but to align its statistical representations toward a narrower range of tasks.[138]

172. The data used for fine-tuning differs from pre-training data in both specificity and intent. Rather than drawing from large text collections, developers typically assemble a smaller corpus that is tailored in structure and relevance to the targeted application.[139] This dataset might consist of medical records, technical manuals, or legislative materials, depending on the model's intended use. For an LLM model to answer a user's questions and requests when prompted, the dataset might consist of questions and answers. A model originally trained to capture every form of language now begins to focus more on certain styles, terminology, and reasoning patterns. Over many iterations, the statistical influence of domain-specific text grows stronger and leads to responses that are more coherent and contextually accurate within that field.

173. An advantage of pre-training on vast amounts of diverse data is flexibility. A single base model can be adapted into multiple specialized versions, each with a different function. One might emphasize analytical reasoning for research purposes while another focuses on concise summarization for productivity tools. Since all variants share the same foundational parameters, they can be updated with less computational cost than would be required for training new systems independently. This modularity enables development of domain-specific

[137] Kumar, K., et al. (2025). LLM Post-Training: A Deep Dive into Reasoning Large Language Models (arXiv:2502.21321). arXiv. https://doi.org/10.48550/arXiv.2502.21321.
[138] Kumar, K., et al. (2025). LLM Post-Training: A Deep Dive into Reasoning Large Language Models (arXiv:2502.21321). arXiv. https://doi.org/10.48550/arXiv.2502.21321.
[139] Liu, Y., et al. (2024). Datasets for Large Language Models: A Comprehensive Survey (arXiv:2402.18041). arXiv. https://doi.org/10.48550/arXiv.2402.18041. ("The domain-specific instruction fine-tuning datasets are constructed for a particular domain by formulating instructions that encapsulate knowledge and task types closely related to that domain.").

[redacted]

language models where precision and reliability are more valuable than generality.

174. Once a model has completed training and is deployed for ordinary use, its weights do not change from one user interaction to the next. More precisely, although pre-training, post-training, fine-tuning, and other development stages adjust the model's parameters, the model's ordinary operation at inference time does not. During deployment, the model generates outputs by applying its existing learned parameters to the user's prompt (and any additional context provided for that response); it does not update those parameters merely by receiving a prompt or generating an output. In that respect, ordinary model operation is distinct from training.

175. Additionally, it is not possible to reliably attribute LLM outputs to particular training data tokens. Existing methods which attempt to do this (e.g., influence functions and attribution techniques)[140] provide only approximate estimates and do not establish direct lineage between training tokens and model outputs in general. Further, even when an output token matches a token seen in the training data, or when an attribution method highlights a particular training token, this may reflect the distributed statistical associations learned from many related training tokens rather than influence from the specific training token itself.

2. Inference

176. As discussed in Section IV.B.6 above, LLMs generate content at inference time by predicting the next token autoregressively, generating each token using as context the tokens already generated, in addition to the provided prompt.

[140] Koh, P. W., & Liang, P. (2020). Understanding Black-box Predictions via Influence Functions (arXiv:1703.04730). arXiv. https://doi.org/10.48550/arXiv.1703.04730; Ruis, L., et al. (2025). Procedural Knowledge in Pretraining Drives Reasoning in Large Language Models (arXiv:2411.12580). arXiv. https://doi.org/10.48550/arXiv.2411.12580; Rosser, J., et al. (2026). Infusion: Shaping Model Behavior by Editing Training Data via Influence Functions (arXiv:2602.09987). arXiv. https://doi.org/10.48550/arXiv.2602.09987.

[redacted]

177. The objective of all model training stages, including pre-training and post-training, is to improve the model's ability to generalize to new and unseen prompts within a targeted domain, rather than toward specific training examples. The resulting outputs of the models are not fixed in advance, but are generated dynamically at inference time from probability distributions learned over the entirety of the data used for training the model. The textual material used in training is not explicitly stored to later be reproduced during normal model operation. Instead, the model generates new text by applying learned statistical relationships to the prompt and context presented by the user.

178. Because a model's parameters encode statistical patterns, a model operating without access to external data sources may provide factually accurate information acquired from statistical associations in the training corpora, but has no mechanism for confirming the factual accuracy of its generations. Models may generate confabulations where the output is syntactically and semantically well-formed, creating an internally coherent but false statement. Models may also generate other types of inaccurate statements such as inventing citations or applying a learned pattern beyond its valid domain. All such types of generations have colloquially come to be referred to as "hallucinations" but the anthropomorphic nature of this term is misleading. These types of generations are known aspects of an ungrounded system, and explain the increasing importance of retrieval-augmented generation, which allows the system at inference time to retrieve potentially relevant, current information from external sources and provide that material as temporary context for generation. As described in Section IV.B.11), this can improve factual accuracy and relevance, but the retrieved material remains external to the model's parameters and does not alter or persist in the model itself.

D. LLM External Data Access

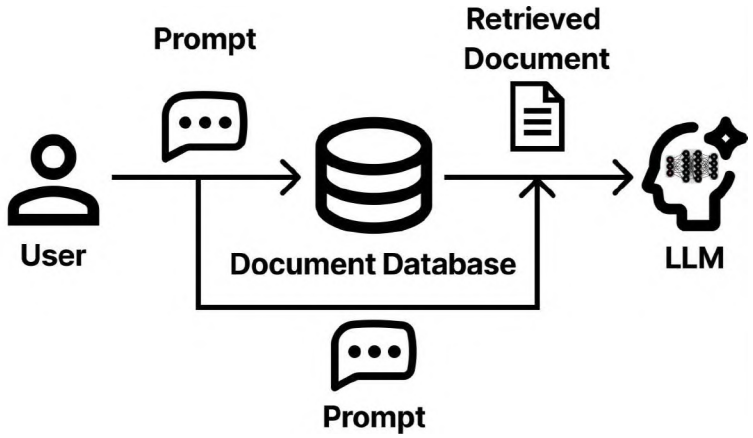
1. Grounding

179. Grounding is the method of combining a pre-trained language model with an information retrieval ("IR") system, which can search external data sources as the

[redacted]

LLM generates text during inference.[141] That is, rather than relying solely on the knowledge encoded in its parameters, the model can access a separate collection of data via a retrieval system and use retrieved material as part of the prompt context for a particular response. At inference time, the LLM can analyze, synthesize, and summarize the textual material in response to a user's query. This section first describes the basic grounding pipeline, then explains the main retrieval methods used in modern grounding systems, and finally discusses how grounding operates when integrated into large-scale products and web-connected systems. Microsoft witnesses address the Copilot product's retrieval and orchestration of grounding material; my discussion addresses the technical operation of the LLM once retrieved material is supplied as context.

Figure 22: Grounding



180. A grounding system functions as a multi-step process, as demonstrated in Figure 22. When a user enters a prompt, the system first transforms that prompt into a query that is suitable for retrieval. The IR system then searches a database for documents or other sources that relate closely to the user's query. The top results are converted into tokens and attached to the original prompt as part of the model's actual input. The LLM reads this combined context – both the user's question and the retrieved passages together – and generates a response. The

[141] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). pp. 233–234.

[redacted]

final output is therefore influenced by how the model interprets the retrieved material relative to its own internal knowledge.[142]

181. A key feature of grounding is the separation between retrieval and text generation, which are separate processes that operate independently but in coordination. The retrieval component determines which external material is relevant to a given prompt; the generation component (the LLM) produces a response based on that material and its own learned parameters. As is the case with general model inference, the underlying LLM is not altered, retrained, or updated by the retrieval process. Retrieved documents are temporary inputs that apply only to the current prompt and are discarded after the response is generated. Each new prompt triggers a fresh retrieval, and material retrieved for one prompt does not persist into subsequent interactions.

182. In practice, grounding systems differ primarily in the nature of the external information sources they query. When an LLM powered by grounding processes a prompt, it may retrieve text from a variety of external sources, such as curated document collections, enterprise knowledge bases, company repositories, or search services that provide publicly available web content. Depending on the application, these systems may consult verified internal materials, domain-specific documents, current information, or a combination of these.

183. Regardless of the type of data collection, retrieved passages are provided to the LLM only as temporary context for a given response and do not alter the model's internal parameters or persist beyond the current request. More specifically, grounding does not modify the model's learned parameters, incorporate the retrieved material into those parameters, or make the retrieved documents part of the model itself. Rather, grounding supplies additional text for the model to consider at inference time for that response alone, after which the model discards it. This external material is not generally retained by the LLM.[143]

---

[142] Jurafsky, D., & Martin, J. (2025). Speech and Language Processing (Third Edition draft). p. 246.
[143] Lewis, P., et al. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (No. arXiv:2005.11401). arXiv. https://doi.org/10.48550/arXiv.2005.11401.

[redacted]

184. Continuing the example used in Section IV.B.6), in response to the prompt "What is typical fall weather in New England?" the model might generate text such as, As the weather becomes colder and the days grow shorter, trees in New England turn bright colors, beginning in late September.

185. The model might then continue to describe recent trends, grounding its response in recent contextual specificity: However, in recent years the fall colors can continue into November, and the 2025 autumn was the 3rd warmest on record.

186. This last sentence might not be based on any factual information learned from the training text. Instead, it could be determined "on the fly" by making a query to weather.com, and summarizing the results that were found. In that respect, grounding allows the model to answer the user's question by analyzing and summarizing external text retrieved at inference time, rather than relying only on information internalized during training.

187. As described in the discussion of RLHF above, the post-training alignment process can be used to shape the model's behavior during grounded generation—including by penalizing outputs that reproduce retrieved text too closely. In practice, developers use RLHF and other alignment techniques to encourage the model to summarize, synthesize, and cite retrieved information rather than reproduce it verbatim. These alignment measures function as guardrails that complement the architectural separation between retrieval and generation: even where the model has access to retrieved text as input context, the model's generation is shaped by its training to produce original responses that draw on, rather than copy, that material.

188. The retrieval step can be implemented in different ways, depending on the source being searched and the type of matching the system is designed to perform. Some systems retrieve from curated, static document collections using semantic search, which identifies relevant material based on the similarity of numerical vector representations rather than exact keyword matching. Other systems, including web-connected products like Copilot, issue search queries to a web search index and retrieve content from publicly available web pages in real time. In some

[redacted]

implementations, both methods operate together: semantic search handles retrieval over private or structured data, while web search queries handle open-domain information. Regardless of the retrieval method used, the retrieved material is provided to the model as temporary prompt context and does not alter the model's parameters.

189. LLM outputs generated with grounding use retrieved documents as transient, inference-time signals to condition the model's generation, resulting in newly created token sequences that reflect task-specific generation rather than the exact nature of the retrieved texts. When integrated into products such as Copilot, grounding connects the model's language capabilities with external information sources, enabling the system to generate responses that incorporate current, verifiable information retrieved at the moment of the user's query. The output of this process is a newly generated sequence of tokens, produced sequentially rather than retrieved as a completed passage; it is a new generation informed by the retrieved material.

190. The retrieved material that a grounding system supplies to the model is constrained by the model's context window. As a matter of standard engineering practice, grounding systems typically supply portions or processed representations of retrieved content, meaning text that the search infrastructure has segmented, embedded, and indexed, rather than necessarily supplying each retrieved source in full as originally published; supplying the full text of every retrieved source could overflow the context available to the model. At inference time the model is not itself pulling articles from the Internet; it is supplied excerpts or processed representations of search-result content assembled by the retrieval system.

191. In sum, in a grounded interaction the retrieval system supplies selected external material to the model as temporary context together with the user's prompt. The LLM processes the tokens supplied within its context window, including the user's prompt, any prior conversation, and the grounding material, and generates the user-facing natural-language response token by token, with each token

[redacted]

sampled from a probability distribution conditioned on that combined context and on the model's learned parameters. The grounding material is an input to the generation process; it is not itself the user-facing response, and the response is produced sequentially rather than retrieved as a completed passage. The presence of text, a URL, or a domain in an internal grounding field accordingly establishes that material was retrieved and processed by the system; it does not by itself establish that the same material appeared in the response shown to the user. Some overlap between retrieved text and a generated response can occur, particularly for short factual spans, as an expected byproduct of summarizing retrieved material in natural language; whether source expression actually appeared in a response is determined by examining the response itself, as the output analyses described below do.

V. CHAT LOG DATA OVERVIEW

A. Copilot

192. This section provides a brief overview of the Microsoft product relevant to the Chat Log Data analyzed in this declaration. It is my understanding that Microsoft witnesses will provide a more detailed account of the development and operation of that product. My description here is limited to the technical context necessary for understanding my analysis of the Chat Log Data.

193. I was asked to study the Chat Log Data for the consumer-facing conversational AI assistant that Microsoft initially introduced on a limited basis in February 2023 as Bing Chat (later rebranded as Microsoft Copilot; referred to in this declaration as "Copilot") and made generally available to the public from December 2023 through September 2024.

194. Between February and November 2023, access to the product was limited – first through a waitlist and then through a requirement that users sign in with a Microsoft account. The product became generally available to the public without

[redacted]

access restrictions beginning in December 2023.[144] The product underwent a major overhaul in October 2024, as I understand that Copilot was fully deprecated and replaced by a new consumer Copilot on October 1, 2024.[145] Copilot used OpenAI's GPT family of large language models, including variants of GPT-3.5 Turbo, GPT-4, and GPT-4 Turbo.

195. The Chat Log Data analyzed in this declaration corresponds specifically to Copilot during the period of general public availability –– from December 2023 through approximately September 2024. The data does not include conversations from the earlier preview or limited-access periods. That exclusion is appropriate because the objective of my analysis is to assess general usage patterns of the free, consumer-facing version of the product during the period when it was broadly accessible to the public.

B. Data Storage and Structure

196. The Chat Log Data is stored in Azure Data Lake, Microsoft's cloud-based data lake service that is designed to hold large volumes of data in native file formats. The Chat Log Data is approximately 13 petabytes in size for the relevant date range,[146] which is akin to 6.5 trillion pages of text printed on standard 8.5x11 paper that, if stacked vertically, would span the distance from the Earth to the Moon and over halfway back.[147]

197. [redacted]

---

[144] Microsoft Copilot is now generally available. (2023, December 1). Microsoft Copilot Blog . https://www.microsoft.com/en-us/microsoft-copilot/blog/2023/12/01/microsoft-copilot-is-now-generally- available/.

[145] Hurst Decl. Ex. 99, Nov. 5, 2025 30(b)(1) Deposition of Jordan Ribas at 19:7-20:11; Hurst Decl. Ex. 1, Nov 18, 2025 30(b)(6) Deposition of Elbio Abib at 39:21-40:7, 59:14-20; *See also* Microsoft Corporate Blogs. (2024, October 1). An AI companion for everyone . The Official Microsoft Blog. https://blogs.microsoft.com/blog/2024/10/01/an-ai- companion-for-everyone/.

[146] file_lister.sh recursively lists all files within the specific directories for the logs and outputs the file name and the size of the file in terabytes. file_sizes.sql sums the sizes of the individual files to get the total data size.

[147] Assuming each 8.5x11 page fits 2KB of information, which is about ~0.004 inches thick. 13PB / 2KB = 6.5 trillion pages. 6.5 trillion pages × 0.004 inches per page = 26B inches = ~410,000 miles. 410,000 miles ÷ 238,855 miles (average distance to the moon) = ~1.71.

[redacted]

[redacted][148]

198. Given these different categories of information, the data architecture is organized into a set of views, each offering a different way to analyze the Copilot interactions. A view defines how the raw data is organized and structured for analysis. Views do not introduce new data; instead, they curate existing event data into consistent, searchable representations that align with common analytical questions.

199. [redacted]

200. Prompt and response interactions in the data are defined by turns, where a turn represents a single exchange between a user and Copilot. A turn begins when a user submits a prompt and ends when Copilot generates a response. Conversations are formed when multiple turns occur in sequence, with each turn building on the context established by prior exchanges. [redacted]

C. Data Fields

201. I, along with members of the support team working at my direction, analyzed the data fields to determine the relevant fields and develop a methodology for providing those fields in a manner allowing Professor Christopher Bail ("Dr. Bail") and his support team to organize conversations chronologically and categorize the intent of the users. The data fields selected in my sample are summarized below. A complete list of all 20 fields from the [redacted] used for this analysis are listed in Appendix B to this

[148] Hurst Decl. Ex. 186, MSFT_AI_MDL_001115484 at -484.

[redacted]

declaration. My understanding of these views is based on my review of the Microsoft [redacted] documentation and reviewing exemplar data logs with members of the support team working at my direction. To confirm my understanding of these views, I also had a conversation with an engineer from Microsoft.[149]

a. [redacted]

b. [redacted]

c. [redacted]

d. [redacted]

e. [redacted]

VI. METHODOLOGY FOR CHAT LOG DATA SAMPLE SELECTION

A. Objective

202. To support downstream coding of conversations in the Chat Log Data, given its enormous scale, a sample must be taken that is representative of the population. Sampling is a common technique applied in statistical sciences when working

[149] Hurst Decl. Ex. 186, MSFT_AI_MDL_001115484 at -488; Meeting with Chauncey Chen, a Microsoft Software Engineer overseeing the Copilot Data as facilitated by Counsel.

[redacted]

with extraordinarily large datasets, as in this matter. Further, I understood from Counsel that the analysis of usage patterns was to be conducted by a team of human labelers called "Coders" who would manually categorize the conversations in the Chat Log Data. This necessarily required a sample to be drawn given the size of the population data—it is enormous, totaling approximately 13 petabytes and more than 1 billion conversations over the relevant timeframe of December 1, 2023 to September 30, 2024.[150] Assuming that it would take a Coder 30 seconds to manually categorize a conversation, a team of 5 Coders working in parallel would require approximately 190 years to complete a manual categorization of the conversations. It is obviously infeasible for a small team of people to classify such a large population of data, which therefore requires either some acceptable form of automation for analysis, or a valid method of sampling for human coding.

203. To address the sampling objective, I describe the necessary data preprocessing steps, method of sampling, and size of the sample below.

B. Data Filtrations in Preparation for Sampling

204. This section describes the properties of, and filters applied to, the [redacted] data in order to arrive at a final dataset for sampling. There are seven items discussed: Log Type, Fields, Location, [redacted], User-created conversations, Date, and Non-empty prompts.

205. First, the Chat Log Data only pertains to consumer Copilot logs, and not enterprise Copilot. This is not a filter that is applied, but rather an attribute of the Chat Log Data.

206. Based upon review of documentation describing the data fields and review of sample data logs, the following criteria were applied jointly for the Parties' Experts analyses (see Section VIII.A below):

[150] file_lister.sh recursively lists all files within the specific directories for the logs and outputs the file name and the size of the file in terabytes. file_sizes.sql sums the sizes of the individual files to get the total data size.

[redacted]

a. **Fields:** Filtered to a subset of specific fields which were jointly agreed upon by the Parties for analysis.

b. **Location:** Filtered to keep records only in the United States.

c. [redacted]: Within [redacted], filtered for the presence of [redacted] data, which contain user prompts, model responses, and internal logs.

d. **User-created conversations:** Filtered to retain only conversations initiated by a user through the Copilot interface, as identified by the Source field within the [redacted] in the Chat Log Data.[151]

207. Finally, I applied the following filters in preparation for sampling the data to support the downstream categorization of conversations:

a. **Fields:** I filtered to include specific fields relevant to the downstream user categorization analysis. These fields are listed in Appendix B, which contain information such as the user prompt, model response, and message timestamp, among others.

b. **Date:** I filtered to keep records only within the date range Copilot was generally available to the public (December 1, 2023 through September 30, 2024).

c. **Non-empty prompts:** I filtered to keep only conversations with non-empty user prompts, because categorization cannot take place without a user prompt.

C. Sampling Methodology

208. To capture the general usage patterns for the population of Chat Log Data from the full launch of Consumer Copilot to the depreciation of Consumer Copilot, adopting a simple random sampling ("SRS") is most appropriate. I applied this SRS to conversation IDs (rather than, for example, to individual message turns)

[151] Internal systems utilized the same logging infrastructure to log conversations. However, these conversations were entirely system logging and did not represent an actual interaction with a user.

[redacted]

in order to preserve the entire message turn history associated with a given conversation.

209. Simple random sampling gives each conversation an equal probability of being selected. Therefore, if the sample is sufficiently sized, it will be representative of the overall usage patterns in the population. This helps avoid unintentionally oversampling rare or atypical cases that are not reflective of general usage patterns, and further limits unintended biases in the sample that would prevent it from being representative of the population. In comparison, other sampling techniques, such as cluster sampling or stratified sampling, partition the data into groups before selecting a subset, and require greater care or additional information to prevent the introduction of bias in the data selection.

210. Additionally, the SRS methodology I employed is robust and repeatable. I ran a mathematical function, called a hashing algorithm, on each unique conversation identifier, generating a set of characters using a "seed" value. I then sorted the resulting hashed values and select the determined number of conversations as described in the subsequent section, which maintained the random behavior and also enables reproducibility.

211. With each conversation viewed as a playing card, this methodology can be seen as randomly shuffling all billion cards (one for each conversation) and then taking the top N cards as the random sample. I determined an appropriate sample size N using statistical analysis described in the following section.

212. The details for how I employed SRS and the accompanying scripts used to implement this sampling strategy can be found in Section VIII below.

D. Determining Sample Size

213. The determination of sample size is fundamental to selecting a random sample. In the present setting, it was important to determine how many conversations were needed to reliably characterize the general usage patterns for the chosen population of Copilot Chat Log Data. I describe a conservative approach below to determine the size of the sample, which brings together the notions of statistical

[redacted]

confidence and numerical precision. This analysis is based on a primary guiding question, and checked using two supplementary analyses:

a. **Guiding Question for Sample Size:** What sample size is sufficiently large to yield appropriate accuracy in the downstream usage analysis, in terms of confidence intervals, coverage, and numerical precision?

b. **Supplementary Analysis, Criterion A:** What is the chance that, if an additional conversation were sampled, it would fall into a previously unseen category of conversation?

c. **Supplementary Analysis, Criterion B:** How closely does the number of categories seen in the sample match the number of categories in the population?

1. Applicability of Power Law Distributions

214. The following discussion of missing mass and confidence intervals makes no assumption about the parametric form or "shape" of the distribution of categories. However, power laws are ubiquitous in many types of data, and are widely observed for distributions of data sampled from online sources.[152] In a power law distribution, a few categories or types comprise most of the probability mass, but there exists a long "tail" of categories with relatively small probability mass.[153] A classic example of a power law is the distribution of city sizes. In many countries, if cities are ranked by their population, the population of a city is roughly proportional to 1 divided by its rank.[154] This is often called Zipf's law.[155] It means the largest city is about twice as big as the second-largest, three times as big as the third-largest, and so on.

[152] Mahanti, A., et al. (2013). A Tale of the Tails: Power-laws in Internet Measurements. IEEE Network , 27 (1), 59–64. https://doi.org/10.1109/MNET.2013.6423193.
[153] Zipf's law. (2025, December 15). Britannica. https://www.britannica.com/topic/Zipfs-law.
[154] Zipf, G. K. (1949). Human Behavior And The Principle Of Least Effort. Addison-Wesley Press, Inc. http://archive.org/details/in.ernet.dli.2015.90211.
[155] Zipf's law. (2025, December 15). Britannica. https://www.britannica.com/topic/Zipfs-law.

[redacted]

215. Natural language contains many well-cited examples of Zipf's law. In this case, if the words in a large body of text are ranked by how often they appear, the frequency of a word is roughly inversely proportional to its rank.[156]

216. Prior studies of the contents and uses of LLMs showed similar distributions with characteristics of power laws. For instance, in a study published by Anthropic, researchers analyze the usage patterns of Claude, its AI model. These categories, when plotted by appearance, appear to follow a Zipf distribution.[157] Microsoft has also published literature studying the usage of Microsoft Copilot, categorizing conversations in a similar manner. When plotted as a fraction of total activity, these categories also appear to follow a Zipf distribution.[158]

217. A "truncated" Zipf distribution, that is, a Zipf distribution derived by restricting the domain of possible values, has a probability mass function:

$$p(k) = \frac{k^{-\alpha}}{\sum_{k=k_0}^{k_1} k^{-\alpha}}$$

218. Where:

a. α represents the shape parameter. This controls how steeply the frequency of the distribution decreases with rank. As the shape parameter increases, the distribution decays more quickly; as the shape parameter decreases, the tails of the distribution become heavier (meaning the distribution has a larger probability mass in the tail).

b. k_0 represents the lower truncation point, which can be useful to control the mass in the most frequent ranks.

c. k_1 represents the upper truncation point. This is used to limit the distribution to a relatively small number of types.

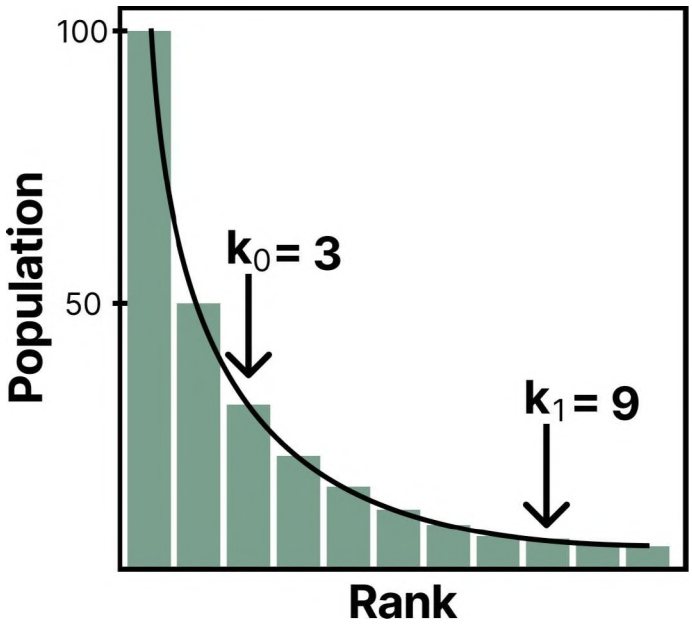
---

[156] Zipf's law. (2025, December 15). Britannica. https://www.britannica.com/topic/Zipfs-law.
[157] Tamkin, A., et al. (2024). Clio: Privacy-Preserving Insights into Real-World AI Use (arXiv:2412.13678). arXiv. https://doi.org/10.48550/arXiv.2412.13678.
[158] Tomlinson, K., et al. (2025). Working with AI: Measuring the Applicability of Generative AI to Occupations (arXiv:2507.07935). arXiv. https://doi.org/10.48550/arXiv.2507.07935.

[redacted]

219. In **Figure 23** below, I give an example of the distribution in which the population of the largest city is twice that of the second largest, and so on. In this figure, one can imagine using the parameters $k_0 = 3$ and $k_1 = 9$. If these parameters were chosen, the distribution would not include the first two cities, nor would it include any cities after the ninth.

Figure 1: Example of Power Law Distribution



220. Although the statistical analysis I carry out does not in any way need to assume the data follow a Zipf distribution, in my robustness checks detailed in Section VII, I validated the methodology and sample size determination on Zipf distributions, since such distributions can be anticipated to arise in the actual data.

2. Sample Size Determination (Confidence Interval)

221. When estimating from a random sample of n conversations the probability $\hat{p}(c)$ that $\hat{p}(c)$ a conversation exhibits usage category c (for example, "research support"), the standard way of computing a 95% confidence interval is given

[redacted]

below.[159] Such an interval will include the true population proportion $p(c)$ 95% of the time. This provides a statistical guarantee on the accuracy of the estimate.

$$\hat{p}(c) \pm 1.96 \sqrt{\frac{\hat{p}(c)(1-\hat{p}(c))}{n}}$$

222. Here, $\hat{p}(c)$ is the fraction of conversations in the sample that have this category. The Coders will choose C categories, and the goal is to have simultaneous control over all the statistical errors. A common approach is to use a **Bonferroni correction**. A Bonferroni correction leads to a conservative procedure with almost 50% wider confidence intervals.

223. To explain, if the confidence interval targets 95%, or mathematically $1-\alpha$, where $\alpha = 0.5$, then the Bonferroni correction says that for C categories, the new confidence interval for each category should have the more stringent coverage $1-\frac{\alpha}{C}$.[160] Therefore, maintaining the example as before, for 20 categories, the confidence interval would increase from 95% to 99.75%.[161] As a result, the probability that all estimated category confidence intervals contain the true population proportion $0.9975^{20} = .9511$ ($\approx 95\%$).

224. In addition, there could be a requirement that the confidence interval is not too wide, to give a targeted numerical precision for the estimate. The standard way this is done is by selecting a margin of error, represented by Δ, for the maximum allowed error between the estimated population proportion and the true population proportion. A typical value is $\Delta = 0.01$. This targets a confidence interval of $\hat{p} \pm 0.01$. This will be achieved if the sample size n satisfies

$$\text{n} \geq \left(\frac{z_{\alpha,C}}{\Delta}\right)^2 \hat{p}(c)\left(1-\hat{p}(c)\right)$$

---

[159] *Confidence Intervals Involving a Proportion*. (n.d.). Retrieved March 23, 2026, from https://mathcenter.oxford.emory.edu/site/math117/confidenceIntervalsProportions/.

[160] *Bonferroni Correction*. (2026, March 25). Wolfram. https://mathworld.wolfram.com/BonferroniCorrection.html.

[161] For $\alpha = 0.05$ and $C = 20$, $1-\frac{0.05}{20} = 1 - 0.0025 = 0.9975$.

[redacted]

225. This requires n to be larger as $\hat{p}(c)$ gets closer to 0.5. Using this approach, as an example, the sample size required for 60 classes C and a 25% maximum class probability p is 20,936.

226. In Section VII.A**.**, I explain how I conducted robustness checks by varying the parameter C, for the number of classes, from 20 to 100, and the maximum class probability parameter, p from 25% to 50%. This demonstrates that, even under conservative assumptions, a sample size of $n = 30{,}000$ is still sufficient to categorize the Chat Log Data use cases for this level of accuracy and confidence.

227. It is possible that there may be conversations for which the intent cannot be classified. This could arise due to prompts not being in English, a significant portion of the prompt being redacted for personally identified information (PII), or even simply that the intent of the prompt may be unclear. To account for such cases, I provided a random sample of 45,000 conversations for classifications. Still, under the conservative assumptions as described in Section VII.A., the number of conversations required for the usage study remains at least 30,000.

3. Supplementary Analysis, Criterion A (Good-Turing Estimate)

228. The prior section discussed how to choose an appropriate sample size to obtain accurate estimates of the population usage patterns from the sample. In addition to estimating the probability of each usage type, it is important to estimate the coverage of the population usage types in the sample, which can be thought of as the chance of missing a usage type. This can be framed by the following question: If a second random sample were to be obtained, what is the chance that a new type of conversation would be observed, which was previously unseen in the first sample? An approach is outlined to answer this question based on the notion of "missing mass" estimation.

229. To make the concepts clear, the problem of missing mass estimation will be described in terms of observing species in a habitat, which was one of the original

[redacted]

applications of the framework.[162] In this example, imagine an avid bird watcher who keeps careful notes on how often he sees each bird species. Over time, he builds up a log showing the total number of birds he has spotted (N), how many times each species has appeared (n_s), and how many different species he has seen at least k times (N_k). The key question missing mass estimation addresses is: based on everything the bird watcher has observed so far, what is the chance that the next time he goes into the field, he will encounter a bird species he has never seen before?

230. This problem is directly relevant to characterizing Copilot conversation logs: a category of Copilot usage plays the role of a bird species, and a specific conversation of that category type plays the role of a sighting of that species. In this way, the missing mass problem is asking for an estimate of the chance that a new type of Copilot usage will be observed in new data, given a random sample of N conversations used for the classification process.

231. An effective and well-known approach to addressing this question is called the Good-Turing estimator,[163] developed by Jack Good and AI pioneer Alan Turing during their code breaking work at Bletchley Park during World War II.[164]

232. The Good-Turing technique is used to estimate the probability of seeing a new category (also known as "missing mass").[165] This gives an answer to the question, "what is the probability that the next Copilot conversation falls into a new, unseen category?" In terms of the notation above, the Good-Turing estimate is given by the ratio of the total number of the categories observed exactly once to the total number of observations:

---

[162] Good, I. J. (1953). The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, *40*(3/4), 237–264. https://doi.org/10.2307/2333344.
[163] Good, I. J. (1953). The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, *40*(3/4), 237–264. https://doi.org/10.2307/2333344.
[164] Good, I. J. (2000). Turing's anticipation of empirical bayes in connection with the cryptanalysis of the naval enigma. *Journal of Statistical Computation and Simulation*, *66*(2), 101–111. https://doi.org/10.1080/00949650008812016.
[165] Good, I. J. (1953). The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, *40*(3/4), 237–264. https://doi.org/10.2307/2333344.

[redacted]

$$\widehat{p_0} = \frac{N_1}{N}$$

233. That is,

a. $\widehat{p_0}$ is the estimated probability of conversation categories not seen in the sample; the "missing mass".

b. N_1 is the number of conversation categories that have been seen exactly once.

c. N is the total number of conversations observed (the sample size).

234. For example, suppose 100 conversations have been observed in total, and 5 of the categories occurred exactly once. Under this scenario, it is estimated that 5% of the population of conversations could be from a previously unseen category.[166]

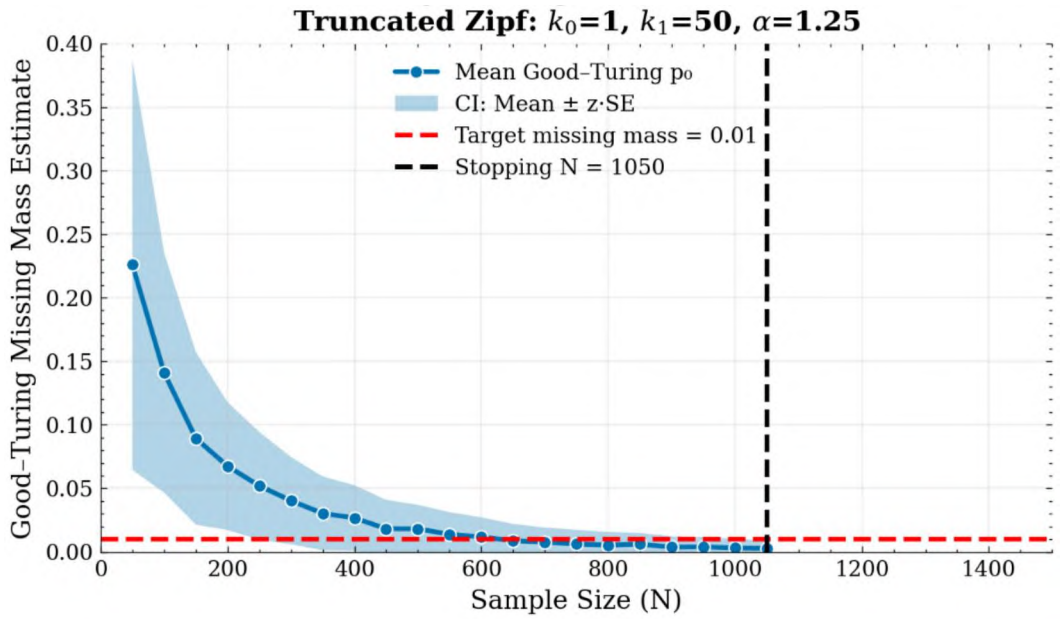
235. Shown below in Figure 24, is an example with data sampled from a truncated Zipf distribution. At a sample size of approximately 1,050, more than 99% of the underlying distribution's probability mass is captured, with 95% confidence. This shows how the Good-Turing estimator closely tracks the true missing mass across the full range of sample sizes. The solid blue line represents the mean Good-Turing parameter p_0, that is, the average probability of the conversation categories not seen in the sample. The lighter blue area surrounding this line represents the 95% confidence interval that bounds this mean p_0. This means if the analysis was repeated many times, the true value p_0 would be contained in this range about 95% of the time. This figure shows the target missing mass of 0.01, which is 1%, is reached after taking approximately 1,050 samples. I explain the additional robustness checks I performed on the Good-Turing estimator using Zipf distributions in Section VII.B.

[166] $\widehat{p_0}$ = N_1/N = 5/100 = 5%.

[redacted]

236. While this example shows the truncated Zipf distribution with parameters α = 1.25, $k_0 = 1, k_1 = 50$ it is important to note that the Good-Turing estimate makes no such assumption about the parametric form (or "shape") of the distribution.[167]

Figure 24: Coverage vs. Sample Size (Good-Turing vs. True Population)



4. Supplementary Analysis, Criterion B (Chao-1 Estimate)

237. In addition to estimating the missing mass, it is common in practices such as ecology to have the goal of estimating the total number of species in a population, since determining the true number of species in a population is not practical. Likewise, determining the true number of conversation types is not possible.

238. The related " Chao-1 " method estimates the total number of species (categories) in a population, including those that were not observed.[168] In other words, Chao-1

[167] Good, I. J. (1953). The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, *40*(3/4), 237–264. https://doi.org/10.2307/2333344.

[168] Chao, A. (1984). Nonparametric Estimation of the Number of Classes in a Population. Scandinavian Journal of Statistics, 11 (4), 265–270; Gotelli, N., & Colwell, R. (2011). Estimating species richness. In Frontiers in Measuring Biodiversity (Vol. 12, pp. 39–54). https://www.uvm.edu/~ngotelli/manuscriptpdfs/Chapter%204.pdf.

[redacted]

provides a statistical answer to the question, "how many categories of conversations exist?"

239. The estimator is as follows:

$$S_{Chao-1} = S_{obs} + \frac{N_1^2}{2N_2}$$

240. Where:

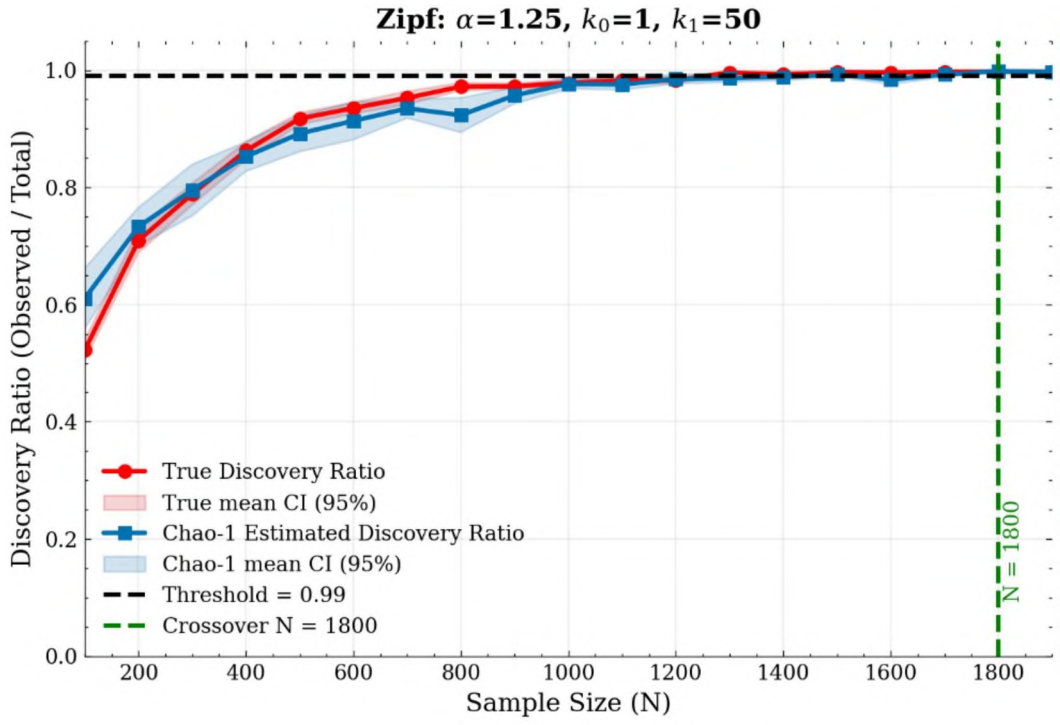
a. S_{Chao-1} is the estimated number of species (i.e., categories)

b. S_{obs} is the number of observed species (i.e., categories)

c. N_1 is the number of conversation categories that have been seen 1 time

d. N_2 is the number of conversation categories that have been seen 2 times

Figure 25: Discovery Ratio vs. Sample Size (Chao-1 vs. True)



241. In **Figure 25**, the solid red line represents the actual discovery ratio of new categories (the number of observed categories divided by the true number of total categories), and the solid blue line represents the Chao-1 estimated discovery ratio (the number of observed categories divided by the estimated number of

[redacted]

total categories). Each line is bounded by a shaded area of the same color that represents a 95% confidence interval, obtained by running the simulation several times. This means if the analysis were repeated many times, the true discovery ratio would be contained in this range about 95% of the time. These lines cross the 99% discovery ratio target with 95% confidence at 1,800 samples. So, in this example, a sample size of 1,800 is sufficient to accurately estimate the total number of categories. In this example, I use a Zipf distribution with parameters α = 1.25, $k_0 = 1$, $k_1 = 50$. I explain the robustness checks on the Chao-1 and Zipf parameters that I performed in **Section VII.C**.

242. The primary driver of the sample size determination, as described above, was the standard confidence level analysis to gauge statistical confidence and accuracy.[169] The standard confidence level analysis accounts for the number of categories and, with increasing category sizes, scales the required sample size proportionally. Using the standard confidence level analysis and very conservative parameters, including using up to 100 categories (which ended up being significantly more than what Dr. Bail used), a sample size of 30,000 was calculated. The final sample of 45,000 conversations was then determined to account for potential challenges in categorizing all logs, such as foreign language and unclear user intent. Thus, this resulting sample size is significantly larger than the calculated requirements.

VII. SAMPLE SIZE ROBUSTNESS CHECKS

243. I conducted additional robustness checks to evaluate whether the calculated sample size required for the conversation review was sensitive to the choice of modeling parameters and to ensure that these choices are conservative. For each type of analysis, confidence intervals on category probabilities, missing mass (Good-Turing), and estimated number of categories (Chao-1), each key parameter is systematically varied across a broad range, and the implied required sample

[169] *Supra* Section VI.D.

[redacted]

size is computed under each configuration. The objective was to determine whether the recommended number of conversations to sample was suitable under alternative, conservative assumptions. These robustness checks are described below. Tables with all results can be found in Appendix C to this declaration.

A. Confidence Interval Robustness Checks

244. Under the Bonferroni formulation of confidence interval, two parameters are varied.

a. **Number of Categories C:** 10 to 100 categories reflects the plausible range of simultaneous categories/estimates and is in line with similar studies.[170]

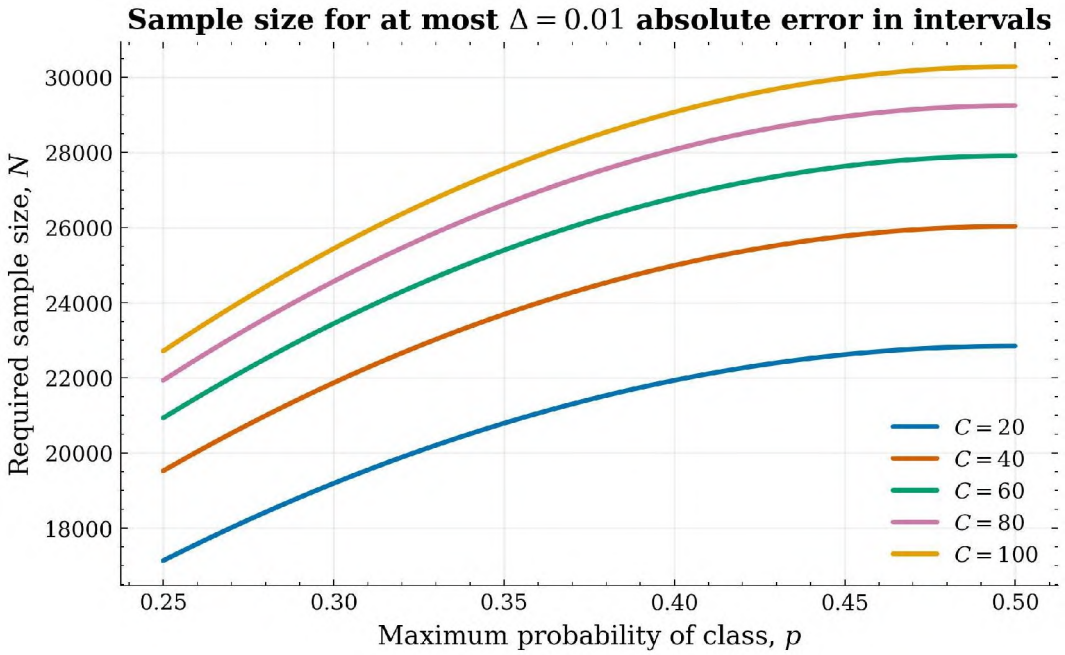
b. **Maximum probability of class *p***: 0.25 (25%) to 0.5 (50%) for classes which are most common in the general use of the tool

245. For each combination of these parameters, the required sample size for conversations was recalculated. This procedure tested whether the sample size determination was driven by any particular assumption of class probability or assumed category prevalence.

[170] Chatterji, A., et al. (2025). *How People Use ChatGPT*. https://www.nber.org/papers/w34255. ; Tomlinson, K., et al. (2025). *Working with AI: Measuring the Applicability of Generative AI to Occupations* (arXiv:2507.07935). arXiv. https://doi.org/10.48550/arXiv.2507.07935. ; Costa-Gomes, B., et al. (2025). *It's About Time: The Temporal and Modal Dynamics of Copilot Usage*. https://arxiv.org/abs/2512.11879v1.

[redacted]

Figure 26: Confidence Interval Robustness Checks



B. Good-Turing Robustness Checks

246. I performed analogous robustness checks for the Good–Turing method of missing-mass estimation, varying five parameters:

a. Zipf Slope parameter $\boldsymbol{\alpha}$: 1 to 1.5

b. Zipf Lower cutoff $\mathbf{k_0}$: 1 to 10

c. **Zipf Upper cutoff $\mathbf{k_1}$**: 10 to 100 to evaluate the effect of alternative truncation thresholds on coverage estimation.

d. Cutoff Threshold of Seen Probabilities $\mathbf{p_0}$: 0.95 (95%) to 0.99 (99%)

247. The sensitivity analyses show that varying the parameters within the specified ranges does not meaningfully change the estimated required sample size. Adjustments to the Zipf slope, truncation bounds, target coverage, confidence level, significance level, precision thresholds, and category counts result in sample size estimates that do not exceed the minimum threshold I determined in Section VI.D. As a minor constraint to maintain reasonable results for the usage analysis, I kept the minimum number of categories (i.e., the number of categories

[redacted]

between k_0 and k_1, inclusive) to be at least 10. The detailed results can be found in Appendix C to this declaration.

C. Chao-1 Robustness Checks

248. A similar robustness check was performed for the Chao-1 estimator for the total number of categories. For this estimator, the parameters that were varied were identical to those varied for the Good-Turing estimator.

a. **Zipf Slope parameter α**: 1 to 1.5

b. **Zipf Lower cutoff k_0**: 1 to 10

c. **Zipf Upper cutoff k_1**: 10 to 100 to evaluate the effect of alternative truncation thresholds on coverage estimation.

d. **Cutoff Threshold of Seen Probabilities p_0**: 0.95 (95%) to 0.99 (99%)

249. Similarly, the sensitivity analyses show that varying the parameters within the specified ranges did not meaningfully change the estimated required sample size. Adjustments to the Zipf slope, truncation bounds, target coverage, confidence level, significance level, precision thresholds, and category counts resulted in sample size estimates that did not exceed the minimum threshold I determined in Section VI.D. As a minor constraint to maintain reasonable results for the usage analysis, I kept the minimum number of categories (i.e., the number of categories between k_0 and k_1, inclusive) to be at least 10. The detailed results can be found in Appendix C to this declaration.

D. Statistical Independence

250. I note that the observations in the original data are not expected to be statistically independent. For example, the usage patterns of the users of Copilot may shift over time. Moreover, if a person uses the tool in a certain way on one day (e.g., to get help with a homework assignment), they may be likely to use the tool in a similar way on the next day (for help with another homework assignment).

[redacted]

251. However, the Simple Random Sampling strategy I employ draws an unbiased and random sample over the entire collection of conversations, and is representative of the overall population of conversations in the collection (comprising more than 13 petabytes over the relevant date range). The methodology involves randomly shuffling the conversations and then taking the top N conversations as the sample. Because the sampling is done without replacement, weak statistical dependencies are introduced. The standard rule of thumb is that if the sample is less than 5% of the population, the weak (negative) correlation between samples can be ignored. With a sample size smaller than 5%, without-replacement sampling can be treated as approximately independent and identically distributed (i.i.d.), and standard results on confidence intervals are valid, and can be applied without modification.

252. If the sample size from the telemetry data is 30,000 out of over a billion conversations, this is less than 0.003% of the population. Therefore, in my opinion, the methodology presented above gives appropriate statistical robustness checks on annotations carried out on the sample, in order to characterize the general usage patterns of Copilot during the time period of interest.

253. I also note that the Good-Turing estimator of missing mass does require independence assumptions. When used for language modeling (to estimate the chance of seeing a new type of word n-gram) or ecology (to estimate the chance of observing a previously unseen species), it is an effective and accurate estimator, even when the observations are statistically dependent. This effectiveness and accuracy carries over to the task of estimating the chance of observing a previously unseen category of usage as well.

[redacted]

VIII. SAMPLE SELECTION PROCESS AND REDACTION FOR PERSONALLY IDENTIFIABLE INFORMATION

A. Selection of Data Agreed by the Parties

254. I was informed by Counsel that the Parties agreed to produce a subset of the data, as described in Section VI.B., to allow for both Parties' Experts analysis. I inspected the corresponding files and confirm they subset the data in a manner that was suitable for my analysis.

255. Specifically, the file metadata_table_final_us_only.usql loads [redacted] stream fields, which underlie the [redacted] and [redacted], within a defined time window agreed upon by the Parties (July 1, 2023 through September 30, 2024) and then consolidates the data so that each record represents a single user interaction ("turn").

256. The file then standardizes timestamps to align all records to the first event time of each conversation and key identifiers, such as client IDs, that are necessary for analysis.

257. The file then filters the data to retain only actual user interactions—specifically, conversations containing messages initiated by users—thereby removing system-only activity.

258. The file then aggregates the data so that each row contains a Chat Message Event and filters the data to conversations in the United States.

B. Random Sampling Implementation

259. The random sampling is implemented in the file final_sample.usql and the final sample was produced as the file final_sample_Dec1_45k_white_listed_masked.csv. Before drawing the random sample, the data was processed according to the filtrations as described in Section VI.B.

[redacted]

260. Then, in order to select the sample, a random, hash-based procedure was used. An MD5 hash[171] value was computed for each conversation using a stable, unique identifier (e.g., ConversationID) combined with the fixed seed string "ai." In practice, this was implemented by concatenating the seed "ai" with the conversation identifier prior to hashing. The MD5 algorithm converts that combined input into a 128-bit hexadecimal value.[172] Although the output appears random, it is deterministic, as the same identifier and seed will always produce the same hash value. This allows for randomness and repeatability in the results.

261. After hash values were generated for all conversations using the seeded input, the full set was sorted in ascending order by hash value. The hashing and sorting process functions as a random shuffle. Because MD5 distributes inputs approximately uniformly across the hash space, sorting by hash produces a pseudo-random permutation of the population.

262. The sample was then defined as the top N conversations in the sorted list. The result is functionally equivalent to drawing N conversations at random from the population. Each conversation has the same probability of appearing within the top N positions (i.e., N / total population), assuming the identifiers themselves are not correlated with variables relevant to the analysis.

263. This approach is reproducible and auditable. Application of the same seed ("ai") and the same hash function to the same set of conversation identifiers will always generate the identical ordering and therefore the identical top N sample. Changing the seed would generate a different, but still deterministic, ordering and pseudo-random sample.

---

[171] Rivest, R. (1992). The MD5 Message-Digest Algorithm (RFC 1321). IETF. https://doi.org/10.17487/RFC1321.

[172] A hexadecimal number is a base-16 numeral system. Rather than using only the digits 0–9 (as in base-10), it uses sixteen symbols: 0–9 and the letters a–f. The letters a–f represent the values 10 through 15, resulting in 16 distinct possible digits for each position.

[redacted]

C. Redaction for Personally Identifiable Information (PII)

264. As instructed by Counsel, Personally Identifiable Information ("PII") is defined as any information that can directly or indirectly identify an individual, including names, addresses, email addresses, phone numbers, and government-issued identifiers, and must not be shared with downstream systems that do not require access to such data. As a result, before any data is shared for conversation classification, it is passed through a redaction system to detect and remove such sensitive information. The redaction for PII is implemented in the folder "Redaction Scripts".

265. In summary, the redaction system: (1) loads input text data, (2) detects PII using Azure AI Language (PII Named Entity Recognition),[173] (3) evaluates detected entities through additional filtering logic, (4) replaces selected PII with a standardized mask ("[REDACTED]"), and (5) creates a new dataset with the redactions. The Azure service uses prebuilt models to identify common categories of PII, including names, locations, organizations, contact information, and numeric identifiers. For instance, if a user prompt is "what is my property value at 123 Sesame Street?", this would be converted to "what is my property value at [REDACTED]?", where the intent of the prompt is preserved but the sensitive information is removed. This process is applied prior to any downstream data processing.

266. Not all fields in the Chat Log Data need to be passed through the PII system, rather, only those fields that may contain free-form or user-generated content require redaction. Specifically, the [redacted] and [redacted] fields are processed for redactions. These fields include user input prompts, model responses, and grounding data used to support responses, all of which may contain embedded PII. Other structured fields, such as numeric IDs and system-generated metadata

[173] What is the Personally Identifying Information (PII) detection feature in Azure Language? (2026, March 30). Microsoft Learn. https://learn.microsoft.com/en-us/azure/ai-services/language-service/personally- identifiable-information/overview.

[redacted]

fields (e.g., timestamps, record identifiers, and routing or logging attributes), are excluded as they are not expected to contain PII.

267. Step (2) above, detecting PII using Azure AI Language, works as follows. For each selected text segment, the system constructs a document and submits batches of such documents to Azure AI Language's recognize_pii_entities endpoint.[174] Azure applies its prebuilt PII Named Entity Recognition models to return detected entities.

268. Step (3) above, applying additional filtering logic, works as follows. Following the PII detection stage, Azure's raw detection output is then evaluated through two filtering layers before redactions are applied based on instruction from Counsel. First, the system applies a set of configuration-driven exclusions for specific entity categories, including Airport, City, Date, Location, Organization, State, and URL[175]. These categories are excluded from redaction. Second, the system enforces a value-based whitelist designed to preserve known non-target entities. This whitelist operates on exact matches: names are only preserved if they correspond to specific approved entries (e.g., designated plaintiffs, businesses, or publications) and are not left visible simply because they are identified as a person. This ensures that ordinary individual names remain subject to redaction. Similarly, email addresses are only preserved when their domain matches an

[174] Azure Text Analytics client library for Python . (2026, March 5). Microsoft Learn. https://learn.microsoft.com/en-us/python/api/overview/azure/ai-textanalytics-readme?view=azure-python.

[175] Also includes Person, PersonType, State, CountryRegion, Continent, DomainName, and Email.

[redacted]

approved domain on the whitelist, rather than being universally allowed.[176] Prior to comparison, candidate values are normalized to ensure consistent matching. If a detected entity matches an entry in the whitelist, it is excluded from the redactions.

269. Step (4) above, applying a standardized mask, replaces all detected PII with the "[REDACTED]" mask. This mask is applied rather than category-specific labels in order to minimize the risk of residual sensitive information, while also maintaining readability and consistency for downstream categorization.

270. The system leverages Microsoft's pre-trained AI model trained to identify and remove sensitive data. As detection is influenced by contextual and formatting factors, accuracy may vary; accordingly, the process prioritizes the removal of data that is likely to be sensitive.

IX. THE OBJECTIVE OF LLM DEVELOPMENT IS GENERALIZATION; PLAINTIFFS MISCHARACTERIZE TRAINING AND MEMORIZATION AS REPRODUCTION OBJECTIVES

271. Plaintiffs' Experts mischaracterize the objective of LLM pretraining when describing (i) next token prediction, (ii) model weights, and (iii) memorization.

[176] Whitelist values enforced by the system include: - Names / Organizations: the athletic; ny times; nyt; new york times; nytimes; wirecutter; new york daily news; ny daily news; daily news; nydn; chicago tribune; tribune; orlando sentinel; sun-sentinel; sun sentinel; san jose mercury-news; mercury news; dp media; denver post; orange county register; oc register; pioneer press; twin cities; mother jones; reveal news; center for investigative reporting; cir; intercept; first look media. - Domains / URL hostnames: nytimes.com; nyt.com; nytcooking.com; cookingnytimes.com; cookingnyt.com; nytimescooking.com; cooking.nytimes.com; newyorktimescooking.com; cookingnewyorktimes.com; theathletic.com; wirecutter.com; sun-sentinel.com; mercurynews.com; ocregister.com; twincities.com; chicagotribune.com; nydailynews.com; orlandosentinel.com; denverpost.com; motherjones.com; revealnews.org; theintercept.com. - Geographic terms: New York; Chicago; Orlando; South Florida; Fort Lauderdale; Broward County; Miami-Dade; Palm Beach; San Jose; San Francisco; Orange County; Twin Cities; Minneapolis; St. Paul; Saint Paul; Denver. - Email domains: nytimes.com; nyt.com; nytcooking.com; cookingnytimes.com; cookingnyt.com; nytimescooking.com; cooking.nytimes.com; newyorktimescooking.com; cookingnewyorktimes.com; theathletic.com; wirecutter.com; sun-sentinel.com; mercurynews.com; ocregister.com; twincities.com; chicagotribune.com; nydailynews.com; orlandosentinel.com; denverpost.com; motherjones.com; revealnews.org; theintercept.com; bayareanewsgroup.com; pioneerpress.com; scng.com; sunsentinel.com; tribpub.com; post- trib.com; southtownstar.com.

[redacted]

A. Next-token prediction is the training signal for learning statistical relationships; Plaintiffs conflate it with an objective to reproduce training text

272. Dr. Goldstein and Dr. Melanie Mitchell frame next-token prediction as though it makes reproduction of training text the objective of LLM training. Dr. Goldstein states that next-token prediction loss is a measure of how well an LLM reproduces its training data, and that this loss is minimized when the model can perfectly reproduce that training text.[177] Dr. Mitchell similarly states that the goal of pre-training is to minimize the loss by adjusting the parameters of the LLM so that its outputs match the training data as closely as possible.[178] This framing confuses the mathematical training signal with the objective of model development. Next-token prediction is the mechanism used to update model parameters during training. It is not an instruction to build a system that stores and later reproduces training text. As I explain above, next-token prediction provides a self-supervised learning signal that allows the model to learn grammar, syntax, factual associations, style, and other statistical regularities across the corpus. The purpose of training an LLM is to learn patterns that generalize beyond the text examples shown during training. Through training on large and diverse collections of text, an LLM learns statistical relationships that can be applied to new prompts and contexts.

273. This distinction matters because a general-purpose LLM that simply memorized the training corpus would not perform well across new prompts, tasks, and context. All machine learning models learn from training examples, but this does not imply that the true objective is verbatim reproduction. A model uses training

[177] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶162; Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 175-76 ("For all of the models we are talking about, they were trained with next-token prediction loss").

[178] Hurst Decl. Ex. 249, Mitchell Opening Report, ¶75; Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 21:01–21:23. ("From a machine-learning point of view, the goal is to learn the distribution that the data comes from, and so we imagine the data as being drawn from a distribution… And so the training data kind of defines that distribution we're trying to model."); *Id.* at 175:06–176:10 ("For all of the models we are talking about, they were trained with next-token prediction loss.").

[redacted]

examples to learn general structure that is useful for predicting outputs for inputs selected from a general population, not simply inputs drawn from the training data. Dr. Goldstein's narrower statement that a base model can predict the next word only in the context of completing articles from its training data is also overstated.[179] An LLM is not limited to simply completing articles that appeared in the training dataset. It generates continuations by applying statistical relationships learned across the training distribution to the prompt and context it receives, including new or user-generated prompts that are unlike any specific document in the training data. Plaintiffs' Experts are correct only in the narrow sense that next-token prediction is the training signal, where at each position in the training text, the model makes a prediction, and the parameters adjust to increase the likelihood of correctly predicting the actual next token. They are incorrect to treat that comparison as making later reproduction the model's purpose or ordinary function, when in fact generalization is the true objective.

274. Plaintiffs' Experts' framing also omits the role of held-out evaluation in model development. Training loss measures performance on examples used to update model parameters, while validation and test loss measure performance on examples withheld from those updates. A model that merely memorized training examples might reduce training loss but fail to perform well on new, unseen data. As I explain above, the generalization ability of a model is evaluated through testing, where the model is presented with data it has not previously seen, and testing provides a more objective measure of performance because the model does not know the correct answers for that unseen data. In statistical terminology, we say that the training loss is a biased estimate of the model's true loss on the general population of inputs. Dr. Mitchell's own description of held-out next-token prediction is consistent with this distinction because it acknowledges that held-out evaluations measure the model's ability to predict

[179] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶173.

[redacted]

text that was not used to train the model directly.[180] Training loss is useful for optimization, but it is not sufficient to determine whether the model has learned broad statistical associations or has simply memorized the training examples. A model can achieve minimal training loss by overfitting to the idiosyncratic details of the training dataset, while the performance on new, unseen data stops improving or worsens. If the purpose of training were simply to reproduce the training corpus, there would be no reason to evaluate performance on examples withheld from the training data used for parameter updates.

275. The following practice-test analogy illustrates the same distinction. A student studying mathematics may take a practice exam, compare their work against an answer key, correct mistakes, and gradually learn methods for solving a class of problems. The same student could instead approach the practice exam by rote memorization, where they associate a question's particular wording, numbers, symbols, or other surface details with a specific answer. Of course, the use of an answer key during practice does not mean the purpose of studying is to simply memorize the answer key. If a student only memorizes the answers to the practice problems, they may perform extremely well when shown those exact problems again, but this does not show they have learned problem-solving skills that generalize. The student may still struggle on a final exam that uses new questions, different values, or different wording. However, a student will perform well on new test problems if they have used the practice problems to learn methods and skills, and can apply these to new problems. Likewise, in machine learning, overfitting training examples is not the same as learning a rule or

[180] Hurst Decl. Ex. 249, Mitchell Opening Report, ¶78; Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 121:21–122:21. ("Q: What did the evaluation show that used The New York Times's Annotated Corpus? A: To the best of my knowledge, it would have been a held-out next-token prediction eval, so it would have shown a single number."); Hurst Decl. Ex. 72, Deposition of Ben Mann at 31:13–33:04, Feb. 20, 2026 ("At that time, a very common type of fine-tuning was to train a classifier on top of a pre-trained model for tasks like sentiment analysis. But for fine-tuning to something like The New York Times, it would make more sense for it to be generative fine-tuning where the model is trying to predict the next token in text that would have come from New York Times articles, probably -- yeah, I think that's a pretty complete explanation.").

[redacted]

structure that can apply more broadly to new data. A model that has very low training loss may have poor performance on new data.

276. As I explain above, properly trained models perform well on unseen inputs because their parameters capture general patterns rather than specific cases. This is true across several different machine learning tasks. In an image classification task, the goal is to correctly label previously unseen images, not to memorize the labels of the training images. Likewise, in language modeling, the goal is not to always perfectly predict the next word or token in the training data, but to accurately model the probability of the next word in new, previously unseen contexts. The LLM scaling laws that I describe above reflect the same principle: they relate predictive performance to the computational resources used to train it, including model size, dataset size, and compute, and evaluate that performance on previously unseen validation or test data. Generalization is therefore the relevant model development objective, not memorization or reproduction of the training data.

277. To reiterate, next-token prediction is the model training signal, while generalization is the model development objective. This objective is reflected in ordinary model development practice. Model developers monitor performance on held-out validation data, which is not used in the training updates that fit the model's weights. If training loss continues to decrease while validation loss worsens, developers treat that divergence as overfitting, not as successful model development. If the objective of LLM training were reproduction of training text, developers would instead continue training so long as training loss kept decreasing, even when performance on validation data worsened. Plaintiffs' Experts' framing is inconsistent with this practice. The same distinction between reproduction and generalization also carries through to inference. As I explain in Section IX.C, LLM outputs are not fixed in advance, but are generated dynamically from probability distributions that are constructed in each new context.

[redacted]

278. The loss computed on training data is the training signal, the quantity used to update model parameters during optimization. It is not the objective of LLM development. The objective function of LLM development is test loss: predictive performance on held-out data the model has never seen, which is the quantitative definition of generalization. The scaling-law methodology reflected in the published literature on GPT-3-era models selects model size, training data, and compute to optimize test loss, and I have every reason to believe that OpenAI and Microsoft, in pretraining the models at issue, used test loss as a proxy for generalization in this standard way. The true goal is generalization, not memorization of the training data.

B. Model weights represent distributed statistical relationships rather than explicitly stored training text; Plaintiffs mischaracterize them as retrievable copies of training works

279. Plaintiffs describe model weights as though they contain copyrighted works, or particular pieces of text, in a stored and retrievable form. Dr. Mitchell states that after pre-training, a model's weights can be thought of as an approximate compressed representation of the text on which the model was trained.[181] The Times and Daily News Plaintiffs' expert Dr. Oren Etzioni similarly claims that LLMs can retain text in a retrievable form within their weights.[182] Dr. Goldstein similarly asserts that copies of Plaintiffs' content are encoded in model parameters.[183] These framings are technically misleading unless they are tied to how model weights actually operate. As I explain above, a trained model contains no copy of any training document, no index or addressing scheme that maps a

[181] Hurst Decl. Ex. 249, Mitchell Opening Report, §IV.C., ¶88; Délétang et al. (2024). Language modeling is compression. In Proceedings of the International Conference on Learning Representations. https://openreview.net/forum?id=jznbgiynus; Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 50:10–52:17. ("I think the model is calibrated to learn from the data, statistical patterns of that data. So whatever sentence structure and paragraph structure is represented in the broad amount of data, the model is incentivized to learn that structure.").

[182] Hurst Decl. Ex. 250, Etzioni Opening Report §VIII.C., ¶152; Liu, X., et al. (2026). Alignment Whack-a-Mole: Finetuning Activates Verbatim Recall of Copyrighted Books in Large Language Models (arXiv:2603.20957). arXiv. https://doi.org/10.48550/arXiv.2603.20957.

[183] Hurst Decl. Ex. 231, Goldstein Opening Report, §XI.B., ¶308.

[redacted]

work to a location in the parameters, and no procedure that takes the parameters and reconstructs a specific work; the weights are numerical parameters whose values reflect statistical relationships learned across the entire training distribution.

280. The Liu et al. paper cited to support Dr. Etzioni's argument makes a narrower point than the storage-and-retrieval claim in Dr. Etzioni's analysis.[184] Liu et al. fine-tuned models using examples in which a short plot summary was paired with the corresponding full book passage, and then tested whether the fine-tuned models would produce verbatim passages when later prompted with similar descriptions. The study illustrates that fine-tuning a model on a task that rewards expanding semantic descriptions into book passages can make the model more likely to generate verbatim text from held-out books when later prompted with similar descriptions. However, this result does not show that an unmodified deployed model without that fine-tuning, under ordinary use, operates as a document repository, or that its weights contain identifiable document records or components corresponding to specific training works. The paper also reports and acknowledges that, before fine-tuning, the models from OpenAI, Google, and DeepSeek produced near-zero verbatim content.

281. Dr. Mitchell's "compressed representation" framing should also not be confused with ordinary data compression. In ordinary compression, data is encoded according to a specified compression method or format and recovered, either exactly in lossless compression or approximately in lossy compression, through a corresponding decoding procedure.[185] A compressed archive still has file boundaries, an addressing structure, and, importantly, a defined procedure for recovering the encoded contents. LLM weights do not operate in this way, nor are they designed to. They are not organized as compressed files, indexed records, or

[184] Liu, X., et al. (2026). Alignment Whack-a-Mole: Finetuning Activates Verbatim Recall of Copyrighted Books in Large Language Models (arXiv:2603.20957). arXiv. https://doi.org/10.48550/arXiv.2603.20957.

[185] Blelloch, G. E. (n.d.). Introduction to Data Compression. Retrieved May 22, 2026, from https://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf.

[redacted]

document-sized containers, and there is no decompression procedure that takes a set of parameters corresponding to a specific training work and reconstructs that work.

282. This also applies to statements that copyrighted content is encoded or retained in a retrievable form within model weights. A training example may influence many parameters, and any given parameter may reflect information from many training examples. This distributed influence is not a document boundary, location, lookup path, or stable set of parameters corresponding to a single work. At most, parameter count speaks to representational capacity. As I explain above, using model parameters to memorize training verbatim is not generally advantageous for predicting new validation data. When training datasets are very large relative to model size, models allocate representational capacity to shared statistical structure across many examples, rather than to information tied to individual passages. The relevant empirical question is whether the trained model behaves as though it retained the Asserted Works verbatim, not whether model capacity makes some amount of memorization theoretically possible.[186]

283. As I describe above, model weights are numerical parameters distributed across a machine learning network, and their values reflect statistical relationships learned during training. Through pre-training, the weights of the model are adjusted in order to represent generalized information about the semantic structure, grammar, style, and other types of co-occurrence patterns underlying the data. These statistical parameters together enable the model to synthesize natural, fluent language that has never been written down explicitly before by any person. A single training example may contribute to small, distributed updates to the model's parameters, but those updates are not preserved as an identifiable, discrete copy of the text. A single work is not encoded into one parameter, or into a particular set of parameters, and then remains unchanged so that the model can later look up that parameter to recover the work. Nor does the theoretical

[186] Morris, J. X., et al. (2025). How much do language models memorize? (arXiv:2505.24832). arXiv. https://doi.org/10.48550/arXiv.2505.24832.

[redacted]

capacity of a model's parameters show that any particular work is actually stored in this manner. Weights encode statistical relationships learned across the training distribution, some of which may support instances of memorized continuations of text, but that empirical possibility is different from a document look up architecture.

284. During inference, when a model receives a prompt with context, it computes a probability distribution over possible next tokens, selects a token through the decoding process, and then repeats that process using the expanded context. Each generated token changes the context for the next prediction. As I explain above, each new prediction depends on all of the prior tokens, up to a maximum context window size, which is why the output is produced sequentially rather than retrieved as a completed passage.

285. This token-by-token generation process is why LLM outputs are dynamic rather than fixed. Even when a prompt constrains the continuation, the model is generating a sequence through repeated probability calculations. Inference-time settings, context, and controls can further influence output token selection and the response ultimately returned to the user. These include temperature and decoding or sampling strategy, hidden application-level instructions and system prompts,[187] retrieved grounding information, and guardrails or output-filtering controls.[188]

286. By the nature of the dynamic generation process, LLMs can respond to prompts with text that has no matching passage or document in the training data. For example, when prompted, "Provide a recipe for chicken parmesan in haiku style," an LLM may respond with, "Chicken pounded thin Egg, crumbs, sauce, and cheese now bake Dinner turns golden."

287. The output combines recipe conventions, poetic form, ingredient relationships, and ordinary language patterns into a new, unique response. Plaintiffs' Experts'

[187] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). Pg. 152-157.
[188] Dong, Y., et al. (2024). Building Guardrails for Large Language Models (arXiv:2402.01822). arXiv. https://doi.org/10.48550/arXiv.2402.01822.

[redacted]

storage-and-retrieval framing does not explain this behavior, as there is little reason to expect a trained model to contain and retrieve a preexisting document matching that specific prompt.

288. This declaration illustrates above the same point with a novel word-transformation task. When a model applies a rule supplied in the prompt to transform "language" into a new output, the rule reflects abstract pattern induction to a previously unseen input, rather than memorization or reproduction of a stored training passage. The model's weights reflect statistical relationships learned across the training distribution, and outputs are generated by applying those relationships to the prompt and context presented to the model.

289. Under the same generation process, an LLM may sometimes generate "regurgitated" text, but this does not imply that the model is a hybrid architecture that includes a hidden document repository for particular edge cases. Since LLMs continually generate text by applying learned statistical associations, in some circumstances, those associations may make a particular sequence of words highly probable, especially if the words and associated concepts are famous, repeated, or distinctive. For example, a prompt about the U.S. Constitution may lead a model to generate the preamble in its response, "We the People of the United States, in Order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common defence, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the United States of America," because that particular string of words is strongly associated with the subject. Such an association may arise because the Constitution and its text are famous and widely discussed across many sources, including textbooks, websites, articles, court

[redacted]

materials, and educational resources.[189] These sources may quote, paraphrase, explain, and contextualize the Constitution in many different ways, causing certain concepts and phrases (i.e., the preamble) to become highly associated with prompts about the Constitution. Similarly, third-party copies of Plaintiff works could be on the open web or in article excerpts hosted on other sites. The model may also generate a unique, original explanation of the Constitution's drafting, ratification, structure, amendments, and significance. Memorization is better understood as an unintended, and most of the time undesired, edge case of probabilistic generation.

290. One cannot point to a copy of any particular training work inside a trained model. The parameters of a trained LLM are billions of numerical values shaped by optimization over the entire training corpus; it is simply not possible to locate any particular piece of training data in the model parameters. Nor is there any operation on the parameters that extracts or reads out stored training text: the only mechanism by which a trained model produces text is inference, that is, next-token prediction.

291. For the same reasons, characterizations of a trained model as a compressed copy or "zip file" of its training data are technically misleading. A zip archive is produced by a compression algorithm that guarantees exact reconstruction of its input. A trained language model contains no such archive and supports no such reconstruction: there is no "uncompress" function built into the model, and no "unzip" function that reveals the training data, either with or without loss.

[189] The Constitution of the United States: A Transcription . (2025, December 19). National Archives. https://www.archives.gov/founding-docs/constitution-transcript; Constitution of the United States . (n.d.). Constitution Annotated. Retrieved May 23, 2026, from https://constitution.congress.gov/constitution/; Cornell Law School. (n.d.). U.S. Constitution . Legal Information Institute. Retrieved May 23, 2026, from https://www.law.cornell.edu/constitution; The United States Constitution . (n.d.). National Constitution Center. Retrieved May 23, 2026, from https://constitutioncenter.org/the-constitution.

[redacted]

C. Memorization can occur in limited circumstances; Plaintiffs overstate its role in ordinary LLM operation

292. Three distinct concepts should not be conflated: a model's retention of facts and statistical patterns; memorization of specific expression from a training example; and reproduction of that expression in a user-facing output. Only the last is regurgitation, and the first does not entail the second or third.

293. Dr. Goldstein states that memorization is driven by factors such as model capacity, duplicated training examples, prompt context,[190] next-token prediction,[191] training order,[192] and model size.[193] Dr. Mitchell similarly identifies increasing model size, output quality,[194] and repetition of text during pre-training.[195] I do not dispute that LLMs can, under specific circumstances, generate text that is the same, or nearly the same, as excerpts of the training data. For example, I described above how targeted fine-tuning can increase the chance a model will output training text when explicitly prompted to. As I explain above,

---

[190] Carlini, N., et al. (2023). Quantifying Memorization Across Neural Language Models (arXiv:2202.07646). arXiv. https://doi.org/10.48550/arXiv.2202.07646; Kandpal, N., et al. (2022). Deduplicating Training Data Mitigates Privacy Risks in Language Models (arXiv:2202.06539). arXiv. https://doi.org/10.48550/arXiv.2202.06539.

[191] Carlini, N., et al. (n.d.). Extracting Training Data from Large Language Models (page 2: Training Objective).

[192] Tirumala, K., et al. (2022). Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models (arXiv:2205.10770). arXiv. https://doi.org/10.48550/arXiv.2205.10770; Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 179. [redacted]; Hurst Decl. Ex. 211, OPCO_MDL_001876595 at -596 (September 2022 chat thread where OpenAI employee Vik Goel says [redacted]).

[193] Hurst Decl. Ex. 231, Goldstein Opening Report, §XI.A.; Tirumala, K., et al. (2022). Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models (arXiv:2205.10770). arXiv. https://doi.org/10.48550/arXiv.2205.10770; Carlini, N., et al. (2023). Quantifying Memorization Across Neural Language Models (arXiv:2202.07646). arXiv. https://doi.org/10.48550/arXiv.2202.07646; Hurst Decl. Ex. 84.B OPCO_MDL_001687062 at -066.

[194] Huang et al., Demystifying Verbatim Memorization in Large Language Models, 10711–10732; Hurst Decl. Ex. 185, MSFT_AI_MDL_000950798 at –799; Id. at –804.

[195] Hurst Decl. Ex. 249, Mitchell Opening Report, §IV.F.3.; Leveling, J., et al. (2024). Evaluation of document deduplication algorithms for large text corpora. In G. Nicosia, V. Ojha, S. Giesselbach, M. P. Pardalos, & R. Umeton (Eds.), Machine learning, optimization, and data science: 10th International Conference, LOD 2024, Castiglione della Pescaia, Italy, September 22–25, 2024, revised selected papers, Part I (pp. 390–404). Springer. https://doi.org/10.1007/978-3-031-82481-4_27.

[redacted]

memorization is a known phenomenon in LLMs, particularly where text appears repeatedly in training data, contains unusual or identifying token sequences, or is elicited through extraction-style prompting. I describe memorization as bounded rather than dominant, and as dependent on the composition of the training data and how the model is probed.

294. Another factor that can affect whether memorized text is elicited is the decoding method used at inference. Temperature is a decoding parameter that reshapes the probability distribution over possible next tokens before sampling.[196] When temperature is low, the probability of higher-probability tokens increases and the probability of lower-probability tokens decreases, which reduces variability in the output. As temperature approaches zero, decoding approaches greedy decoding, where the model selects the highest-probability next token (according to the model) at each step. If the prompt itself makes a training-text continuation unusually likely, a low-temperature setting can make the model more likely to select the continuation. Fine-tuning or in-context examples that are directed at completion-style behavior may also make such outputs more likely. This is not the same as ordinary, open-ended model use.

295. Regarding repetition of training data, the provenance and distribution of text are also relevant. A passage from a popular article may appear in multiple places across the web, including social media, quoted excerpts, aggregators, previews, and other websites,[197] and these duplicative appearances can reinforce the same sequence or near-sequence of text during training. Consider the example with the U.S. Constitution prompt in Section IX.B. This issue may be especially relevant

[196] Jurafsky, D., & Martin, J. (2026). Speech and Language Processing (Third Edition draft). Pg. 156-157.
[197] Kandpal, N., et al. (2022). Deduplicating Training Data Mitigates Privacy Risks in Language Models (arXiv:2202.06539). arXiv. https://doi.org/10.48550/arXiv.2202.06539. ("Most web-scraped datasets are deduplicated at the document level, e.g., if two web pages have the exact same contents, only one is kept in the data. Lee et al. (2021) observe that these datasets still have large-scale approximate and exact sequence-level duplication, e.g., quotes, paragraphs, and advertisements appearing in many web pages.")

[redacted]

for news article titles, leads, or opening passages, which are more likely to be quoted, previewed, linked, summarized, or reproduced across the web.[198]

296. Model capacity can affect memorization risk, but the literature cited by Plaintiffs' Experts addresses narrower conditions than ordinary use. Dr. Goldstein states that larger models have more parameters, which increases their capacity to encode verbatim text and affects memorization risk,[199] and Dr. Mitchell claims that LLMs can memorize large parts of the training data.[200] The studies Dr. Goldstein relies on measure what models do under conditions designed to elicit memorization (i.e., prompting a model with prefixes of its own training data and observing what it emits) and not what models do in response to ordinary user prompts. Tirumala et al. studied exact memorization during training, which is defined as whether a model's highest-probability next token matched the ground-truth token in a training context.[201] Although larger models can memorize training data faster and forget less, the researchers did not analyze whether models store entire works as copies or ordinarily reproduce substantial verbatim text. Carlini et al. likewise found that memorization increases with model scale, training-data duplication, and the amount of training-data context used to prompt the model.[202] The same will be true of n-gram language models, where

---

[198] Lee, K., et al. (2022). Deduplicating Training Data Makes Language Models Better . Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.577. ("For RealNews and LM1B, derived from news sites, we observe that many near-duplicates occur because the same news article appears on multiple news sites with slightly different formatting.")

[199] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶306; Tirumala, K., et al. (2022). Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models (arXiv:2205.10770). arXiv. https://doi.org/10.48550/arXiv.2205.10770; Carlini, N., et al. (2023). Quantifying Memorization Across Neural Language Models (arXiv:2202.07646). arXiv. https://doi.org/10.48550/arXiv.2202.07646; Hurst Decl. Ex. 84.B, OPCO_MDL_001687062 at -066.

[200] Hurst Decl. Ex. 249, Mitchell Opening Report, ¶124; Hurst Decl. Ex. 210, OPCO_MDL_001697816 at –816; Chang, K., et al. (2023). Speak, Memory: An Archaeology of Books Known to ChatGPT/GPT-4. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.453; Liu, X., et al. (2026). Alignment Whack-a-Mole: Finetuning Activates Verbatim Recall of Copyrighted Books in Large Language Models (arXiv:2603.20957). arXiv. https://doi.org/10.48550/arXiv.2603.20957; Aerni, M., et al. (2025). Measuring Non-Adversarial Reproduction of Training Data in Large Language Models ; Hurst Decl. Ex. 209, OPCO_MDL_001682605 at –606.

[201] Tirumala, K., et al. (2022). Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models (arXiv:2205.10770). arXiv. https://doi.org/10.48550/arXiv.2205.10770.

[202] Carlini, N., et al. (2023). Quantifying Memorization Across Neural Language Models (arXiv:2202.07646). arXiv. https://doi.org/10.48550/arXiv.2202.07646.

[redacted]

the parameters of the models are directly interpretable. However, the study measured extractability by prompting models with prefixes from their own training data and testing whether the model exactly emitted the next 50 tokens. These results identify conditions that increase extraction risk, especially duplication and long prompt context. They do not show that LLMs generally allocate parameters to store verbatim copies of works or that ordinary prompts cause reproduction of substantial copyrighted or commercially published works.

297. The studies cited in Dr. Mitchell's analysis are also narrower than Plaintiffs' Experts' framing. The UC Berkeley and Emory paper tested whether ChatGPT and GPT-4 could identify a missing character name from a short book passage.[203] Correctly predicting a character name or learned association is not the same as reproducing the text of a work. As I explain in Section IX.B, Liu et al. concerns targeted fine-tuning and extraction-style prompting, not ordinary LLM operation. Aerni et al. studied non-adversarial reproduction by measuring whether model outputs to benign prompts overlapped with strings of at least 50 characters found on the public Internet, which the researchers used as a proxy for unknown training data.[204] They acknowledge that their overlap metric captures a mix of rare memorized strings, common phrases, idioms, quotations, boilerplate code, and outputs that combine snippets from multiple sources. While the study reports measurable overlap under certain prompting conditions, including occasional long overlaps, it does not establish that LLMs ordinarily reproduce large portions of copyrighted works or that such overlap reflects stored copies of any particular work. More broadly, encoding of facts, names, or associations in model parameters is not the same as memorization of the expressive text used to convey them.

298. Dr. Mitchell also claims that prompts containing a few words from an article can activate parts of the model that capture statistics relevant to the particular

[203] Chang, K., et al. (2023). Speak, Memory: An Archaeology of Books Known to ChatGPT/GPT-4 . Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.453.
[204] Aerni, M., et al. (2025). Measuring Non-Adversarial Reproduction of Training Data in Large Language Models.

[redacted]

writing style of the article.[205] Distinctive prompt language can make continuation of a training example more likely, but that is different from ordinary open-ended model use. Consider the example with the U.S. Constitution prompt I provide in Section IX.B.

299. The fine-tuning literature supports the same bounded view. Dr. Etzioni claims that alignment techniques suppress rather than eliminate the ability to reproduce text and that fine-tuning can reactivate verbatim recall.[206] Dr. Mitchell's analysis likewise cites fine-tuning as evidence that guardrails can be bypassed and models can be made to regurgitate or very closely paraphrase substantial portions of copyrighted books.[207] As I explain in Section IX.B, the Liu et al. paper cited examines targeted fine-tuning and extraction-style prompting, not ordinary LLM operation. At most, the literature shows that targeted fine-tuning and extraction-style prompts can make memorized sequences easier to elicit. This result fits within the memorization conditions I describe above, including adversarial prompting and extraction-style attacks.

300. Plaintiffs' Experts also overstate what can be inferred about the usefulness of retention in certain cases. Dr. Mitchell states that memorization and regurgitation are essential for the types of tasks LLMs are used for, and inevitable

---

[205] Dr. Mitchell claims that if the base model is prompted with the first few words of a newspaper article by the economist and former New York Times columnist Paul Krugman, the prompt will activate parts of the model that store complex statistics capturing Krugman's writing style; that this will cause the model to generate text that follows similar statistics and, in some cases, text that the model has memorized, in the sense that the model's weights will generate verbatim or highly similar text from the training data; and that language models thereby acquire the ability to imitate the language of specific humans along with patterns of expression that are common to many humans. *See* Hurst Decl. Ex. 249, Mitchell Opening Report, ¶89.

[206] Hurst Decl. Ex. 250, Etzioni Opening Report, ¶152-153; Liu, X., et al. (2026). Alignment Whack-a-Mole: Finetuning Activates Verbatim Recall of Copyrighted Books in Large Language Models (arXiv:2603.20957). arXiv. https://doi.org/10.48550/arXiv.2603.20957.

[207] Hurst Decl. Ex. 249, Mitchell Opening Report, ¶124; Hurst Decl. Ex. 210, OPCO_MDL_001697816 at –816; Chang, K., et al. (2023). Speak, Memory: An Archaeology of Books Known to ChatGPT/GPT-4. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.453; Liu, X., et al. (2026). Alignment Whack-a-Mole: Finetuning Activates Verbatim Recall of Copyrighted Books in Large Language Models (arXiv:2603.20957). arXiv. https://doi.org/10.48550/arXiv.2603.20957; Aerni, M., et al. (2025). Measuring Non-Adversarial Reproduction of Training Data in Large Language Models; Hurst Decl. Ex. 209, OPCO_MDL_001682605 at –606.

[redacted]

because they are a direct byproduct of the next-token-prediction training.[208] Dr. Goldstein similarly states that it is desirable for an LLM to exhibit at least some level of memorization and regurgitation.[209] What a model needs in order to be useful is the fact that a particular event occurred, not the specific sentences a journalist used to report it; as the literature recognizes, the reproduction of training-like text does not by itself establish memorization of any particular work.[210] The general objective of pre-training and post-training is to represent and apply factual, stylistic, and linguistic relationships from diverse sources, rather than to preserve particular expressions of an article for later reproduction.

301. Consider the following prompt, for the purpose of illustration: "Explain the Apollo 11 Moon landing and how it happened. Include specific facts about the mission timeline, the astronauts, the spacecraft, the landing site, major mission events, and the mission's return to Earth. Make the answer detailed but understandable for a general audience."

302. A language model may generate text such as, "Apollo 11 was the NASA mission that first landed humans on the Moon. It launched from Kennedy Space Center on July 16, 1969, landed on the Moon on July 20, lifted off from the lunar surface on July 21, and returned to Earth on July 24, 1969. The mission lasted 8 days, 3 hours, 18 minutes, and 35 seconds ..."

---

[208] Hurst Decl. Ex. 249, Mitchell Opening Report, ¶119; Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 183:24–184:12; Hurst Decl. Ex. 212, OPCO_MDL_001900461 at –463 and –468; Hurst Decl. Ex. 208 OPCO_MDL_001644364 at –365.

[209] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶307, Hurst Decl. Ex. 104, Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 183, [redacted]); 184 [redacted]); Hurst Decl. Ex. 84.A, OPCO_MDL_001669609; Tirumala, K., et al. (2022). Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models (arXiv:2205.10770). arXiv. https://doi.org/10.48550/arXiv.2205.10770; Carlini, N., et al. (2023). Quantifying Memorization Across Neural Language Models (arXiv:2202.07646). arXiv. https://doi.org/10.48550/arXiv.2202.07646; Hurst Decl. Ex. 84.B, OPCO_MDL_001687062 at -066.

[210] Morris, J. X., et al. (2025). How much do language models memorize? (arXiv:2505.24832). arXiv. https://doi.org/10.48550/arXiv.2505.24832; Haviv, A., et al. (2026). We Should Separate Memorization from Copyright (arXiv:2602.08632). arXiv. https://doi.org/10.48550/arXiv.2602.08632.

[redacted]

303. The same language model, when prompted with the same prompt, may generate text such as, "Apollo 11 was the NASA mission that first landed humans on the Moon. It launched from Kennedy Space Center in Florida on July 16, 1969, using a Saturn V rocket, the most powerful rocket NASA had built at the time. The crew was Neil Armstrong, commander; Michael Collins, command module pilot; and Edwin 'Buzz' Aldrin, lunar module pilot. Armstrong and Aldrin landed on the Moon, while Collins stayed in lunar orbit in the command module…"

304. Both responses convey common facts about Apollo 11, but they differ in sentence structure, phrasing, detail, and emphasis. As I explain above, via probabilistic decoding methods such as sample-based generation, modern LLMs will express the same facts and information in distinct ways. This does not suggest that an LLM is incapable of reproducing text in particular circumstances, as I describe above, reproducing text is possible at lower temperatures. Instead, it shows that ordinary factual responses can vary substantially in wording order, levels of detail, and organization across different generations, even when they concern the same facts and are generated in response to the same prompt.

305. Modern LLM training practice also reflects deliberate engineering against memorization. Large models are typically trained for approximately one pass (epoch) over the training data, in part because repeated passes over the same data promote memorization rather than generalization. And as the amount and diversity of training data increase relative to model capacity, unintended memorization decreases: the model is forced to generalize.

306. Extraction demonstrations performed at temperature zero do not reflect ordinary model operation. Temperature-zero decoding forces the model to emit its highest-probability token at every step, and running a model this way, particularly in combination with long verbatim prompts drawn from a target text, can be considered an adversarial attack on the model rather than a measurement of ordinary behavior.

[redacted]

X. LLM GROUNDING DOES NOT DEMONSTRATE "COPYING SUBSTANTIAL PORTIONS" OF THE GROUNDING MATERIAL; THE GOLDSTEIN REPORT CONFLATES RETRIEVAL WITH USER-FACING COPYING

307. Dr. Goldstein argues that as part of the grounding process, Microsoft retrieves and copies a substantial portion of the webpage, including article content.[211] Dr. Goldstein then goes on to cite case documents detailing the number of impressions for Plaintiff domains through Bing Web Search and Chat, concluding that each recorded impression likely copied substantial portions of News Plaintiffs' articles and website content.[212]

308. From a technical and logical perspective, this conclusion does not follow. The number of impressions or domain-hit counts do not establish the proposition Dr. Goldstein offers either. An impression records that the grounding system retrieved and referenced a source in forming a response. It does not measure whether, or how much, a Plaintiff's article text was supplied to the model as context, and it does not measure whether any such text outside of the URL appeared in the user-facing output. In other words, grounding means that information from a source may be retrieved, referenced, or provided as context to help formulate a response. It does not by itself establish that the user-facing output contains copied, reproduced, or regurgitated source material outside of the URL.

309. An impression also does not establish output copying. To the extent that an "impression" refers to an instance of Copilot citing, surfacing, or interacting with a domain or search result for the purposes of grounding,[213] this is not the same as a finding that the generated output contained copied article text. Dr. Goldstein's

---

[211] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶416.
[212] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶417-418.
[213] Hurst Decl. Ex. 71, Sathish Manivannan Deposition Tr. 77:16-22 (confirming that the [redacted] field in Microsoft's data refers to [redacted] in discussion of MSFT_AI_MDL_001880566); Hurst Decl. Ex. 194, MSFT_AI_MDL_001880566; Hurst Decl. Ex. 71A, MSFT_AI_MDL_002294702 (confirming that Microsoft can [redacted]).

[redacted]

analysis conflates retrieval, or use of grounding context, with copying in the generated response. A grounded response may be based in part on source context while using different wording, summarizing only high-level details, answering a question without reproducing source expression. A response can use information from a source without copying the source's wording, as illustrated by the example above of a response grounded with weather information. The grounded response uses current weather information retrieved at inference time to answer a question about fall weather in New England by summarizing, rather than reproducing, the retrieved material.

310. Dr. Goldstein's analysis conflates two distinct steps in the grounding process with the premise that each impression reflects copying of a substantial portion of the webpage, including article content. The first is retrieval, in which the grounding system obtains material from a source and supplies it to the model as context for a particular response. The second is generation, in which the model produces the output presented to the user. As I explain above, a key feature of grounding is the separation between retrieval and text generation; the retrieval component determines which external material is relevant to a given prompt, and the generation component produces a response based on that material and its own learned parameters. What the grounding system supplies to the model for a given response is determined by the retrieval method and the fields the system passes as context. The presence of these fields (and the URLs/domains in these fields) establishes that some content is retrieved and processed by the model in forming its response; it does not establish that each impression involves copying a "substantial portion" of a Plaintiff's article. To the extent Dr. Goldstein treats the number of impressions as a proxy for the number of times a substantial portion of a Plaintiff's article was copied, that inference is not supported.

311. The Copilot grounding system fields are described in Dr. Goldstein's source-code analysis, where the returned webpage content includes "context" and "snippet

[redacted]

fields."[214] These fields may support that some content has been retrieved or processed by the model in forming its response, but they do not establish that each impression involved copying any portion, let alone "substantial portions", of a Plaintiff's article. Dr. Goldstein does not quantify how much, if any, of Plaintiffs' article text was included in those fields for each impression, nor does he measure how much of any retrieved text appeared in user-facing output. The impression and domain-hit counts on which Dr. Goldstein relies do not quantify the amount of Plaintiffs' expression supplied to the model as context, and do not establish that Plaintiffs' expression appeared in the user-facing output. Measuring either requires measuring the extent to which Plaintiffs' expression appears in the output and grounding content, not counting impressions. Because the matching counts many disparate fragments across unrelated articles (see Section XIII below), a large hit count does not establish that any substantial portion of any particular Asserted Work appeared in a response.

312. Further, grounding is used to ensure outputs have a factual basis and to reduce the likelihood of hallucinations. The material that LLMs use for grounding is typically largely factual content.[215]

XI. LLM CAPABILITY IS DRIVEN BY BROAD, DISTRIBUTIONAL LEARNING ACROSS DIVERSE CORPORA; PLAINTIFFS OVERSTATE THE UNIQUE FUNCTIONAL IMPORTANCE OF NEWS AND BOOK-LIKE CONTENT

A. High-quality data can support model performance; Plaintiffs conflate data quality with unique functional importance

313. Dr. Etzioni claims that news content is particularly valuable to LLMs because it (1) provides "unusually high-quality text" that is useful for training, (2) contains expressions and events that LLMs rely on to provide accurate and authoritative answers, and (3) contains content such as opinion columns and in-depth analyses

[214] Hurst Decl. Ex. 232, Goldstein Source Code Report, ¶90.
[215] Supra Section IV.D.1.

[redacted]

which provide multiple perspectives that enable LLMs to provide more sophisticated and nuanced answers to topics covered in the news.[216]

314. Class Plaintiffs' expert Dr. Zhao claims books are an important input for high-performing LLMs, and their value is supported by evidence from OAI, MSFT, other developers, and academic research.[217] Dr. Zhao argues that training benefits depend heavily on dataset quality, and that books are generally higher-quality, less repetitive, and more linguistically diverse than typical web text, which makes them more useful per unit of training.[218] Books are also said to be especially useful because they contain long, coherent stretches of text, unlike much web data, which is often fragmented or decontextualized.[219] Dr. Zhao claims that the valuable features of books implied to be are attributes of commercially published copyrighted books specifically, rather than books as a general category.[220]

315. Dr. Etzioni's and Dr. Zhao's statements conflate the idea that high-quality data can improve model performance with the different claim that news or books are uniquely necessary or materially responsible for LLM capability. Establishing

---

[216] Hurst Decl. Ex. 250, Etzioni Opening Report, ¶74. Dr. Etzioni claims that news is particularly valuable for LLMs because, in his view, news is unusually high-quality text that is useful for training LLMs; news contains expressions of key information and events that LLMs rely on to produce accurate and authoritative answers; news content such as opinion columns and in-depth analysis provides perspectives that enable LLMs to provide more sophisticated and nuanced answers to topics covered in the news; and news is an essential source of recent information, enabling LLMs to provide accurate, authoritative, and current responses to queries about recent events.

[217] Hurst Decl. Ex. 251, Zhao Opening Report, ¶14. Dr. Zhao claims that books are a critical input in the development of high-performing LLMs and that the quality benefits from the use of books in LLM training are supported by evidence from OpenAI, Microsoft, their competitors, and third-party research.

[218] Hurst Decl. Ex. 251, Zhao Opening Report, ¶33; Sajith, A., & Kathala, K. C. R. (n.d.). Is Training Data Quality or Quantity More Impactful to Small Language Model Performance.

[219] Hurst Decl. Ex. 251, Zhao Opening Report, ¶67; Simon Rowberry, The Value of Books in the Age of Generative AI Training Data, in Convergence: The International Journal of Research into New Media Technologies, 31(6), pages 1935-1950, 1945 (2025) (quoting Alec Radford, et al., Improving Language Understanding by Generative Pre-Training, OpenAI Preprint (2018)).

[220] Hurst Decl. Ex. 251, Zhao Opening Report, ¶70. Dr. Zhao asserts that the characteristics he describes (contemporary language, editorial polish, genre diversity, sustained narrative coherence) are specific to commercially published, copyrighted works; that public-domain collections such as Project Gutenberg consist predominantly of works published before 1928, written in the language conventions of an earlier era; and that the technical value books provide for training modern LLMs is closely tied to their copyright status, i.e., that in his view it is contemporary copyrighted fiction, not books as an abstract category, that supplies the signal Defendants sought.

[redacted]

that news or book text can improve some outputs does not establish that Plaintiffs' particular works were necessary to the accused models' capability, that they made a measurable marginal contribution beyond the many other sources sharing the same attributes, or that any such contribution is attributable to those works rather than to the broad training distribution. As I describe above, an LLM's general capabilities arise from statistical associations learned across the complete distribution of the training data, and not from any individual text, publisher, or subset of texts.

316. High-quality data can improve model performance, but this does not imply that any single category of high-quality text is uniquely important to LLM capability. The efficiency and efficacy of pre-training depend on the overall mixture of data, including its breadth, diversity, domain coverage, and quality.[221] Diversity in training data allows a pre-trained model to become more useful for a range of applications through post-training.

317. News and books may have useful characteristics and attributes, such as coherence, factual density, recency, editorial quality, and domain coverage. However, these attributes are not unique to news or books. Large scale training corpora commonly include many other source types that contribute similarly important training signals, including encyclopedic material, academic writing, code, educational material, government documents, online discussions, and web text.[222]

318. Dr. Etzioni's analysis cites the "Common Pile" paper written by Kandpal et al, 2025, which documents models trained on a data corpus composed of public domain and openly licensed text, which excludes professionally produced

[221] Liu, F., et al. (2025). QuaDMix: Quality-Diversity Balanced Data Selection for Efficient LLM Pretraining (arXiv:2504.16511). arXiv. https://doi.org/10.48550/arXiv.2504.16511; Gao, L., et al. (2020). The Pile: An 800GB Dataset of Diverse Text for Language Modeling (arXiv:2101.00027). arXiv. https://doi.org/10.48550/arXiv.2101.00027.

[222] Kandpal, N., et al. (2025). The Common Pile v0.1: An 8TB Dataset of Public Domain and Openly Licensed Text (arXiv:2506.05209). arXiv. https://doi.org/10.48550/arXiv.2506.05209.

[redacted]

journalism.[223] Although the models trained on the dataset performed well on many benchmarks, they underperformed on "commonsense reasoning benchmarks." The authors of the paper attribute the gap to the absence of relevant domain coverage, specifically blogs, tutorials, and other content that are not represented well in openly licensed data. Dr. Etzioni's analysis lumps news in with these sources, acknowledging that the study was not designed to test the contribution of news content.[224]

319. The Common Pile paper does not show that news or professionally produced journalism has unique importance or a distinct functional role in LLM training. It was not designed as a news-specific ablation, and it did not compare otherwise-identical models trained with and without news content while holding other domains constant. Rather, the paper compared broader training datasets with different overall mixtures of sources. Although the authors observed weaker performance on certain commonsense reasoning benchmarks, they described the result as possibly reflecting a lack of relevant domain coverage, including domains and topics such as personal blogs, tutorials, hobbies, and sports.[225] This does not establish that news was the missing category, much less that any particular publisher's news content was uniquely necessary. Dr. Etzioni's conclusion goes beyond what the Common Pile paper shows.

320. In general, the research papers in Dr. Etzioni's analysis, which study and describe the importance of high-quality training data,[226] do not show that comparable

[223] Hurst Decl. Ex. 250, Etzioni Opening Report, ¶104; Kandpal, N., et al. (2025). The Common Pile v0.1: An 8TB Dataset of Public Domain and Openly Licensed Text (arXiv:2506.05209). arXiv. https://doi.org/10.48550/arXiv.2506.05209.

[224] Hurst Decl. Ex. 250, Etzioni Opening Report, ¶104; Kandpal, N., et al. (2025). The Common Pile v0.1: An 8TB Dataset of Public Domain and Openly Licensed Text (arXiv:2506.05209). arXiv. https://doi.org/10.48550/arXiv.2506.05209.

[225] Kandpal, N., et al. (2025). The Common Pile v0.1: An 8TB Dataset of Public Domain and Openly Licensed Text (arXiv:2506.05209). arXiv. https://doi.org/10.48550/arXiv.2506.05209.

[226] "Textbooks Are All You Need" (Gunasekar et al., 2023); "Textbooks Are All You Need II" (Li et al., 2023); "Phi-4 Technical Report" (Abdin et al., 2024, Microsoft Research); "LIMA: Less Is More for Alignment" (Zhou et al., 2023, Meta); "Scaling Data-Constrained Language Models" (Muennighoff et al., 2023); "DataComp-LM: In Search of the Next Generation of Training Sets for Language Models" (DCLM) (Li et al., 2024, NeurIPS 2024); "The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale" (Penedo et al., 2024, NeurIPS 2024).

[redacted]

performance could not be obtained from other high-quality sources. High-quality training data is not limited to news articles, and there are many data sources that can support both training and generalization. Openly licensed sources such as Wikipedia, for example, can contribute to domain coverage for LLMs because they provide broad, current, edited, and community-vetted information, which spans many domains.[227] Dr. Goldstein likewise acknowledges that Wikipedia is a high-value, important, and valuable data source.[228] Other examples include Project Gutenberg for public-domain books, arXiv for scientific and technical papers, PubMed Central for scientific and medical publication, Stack Exchange for moderated technical questions and answers,[229] and OpenStax for educational materials.[230] These sources can provide useful attributes such as factual density, explanatory structure, domain-specific terminology, and community or editorial review.

321. Dr. Zhao makes similar statements, focusing on books being a high-quality source of training data. While books may be useful as a training signal since they offer long-form coherence, editorial quality, linguistic diversity, and domain-specific exposition, this does not mean that commercially published copyrighted books are uniquely necessary to general LLM capability. The editorial qualities Plaintiffs associate with their works are properties of text that appear across very large numbers of sources, including uncopyrighted and non-Plaintiff sources. A model learns these features from that broad distribution rather than from any work's copyright status. Other high-quality sources can also contribute to model quality, and an LLM learns from statistical patterns across the full training distribution, not from any single category of works in isolation.

---

[227] Gao, L., et al. (2020). The Pile: An 800GB Dataset of Diverse Text for Language Modeling (arXiv:2101.00027). arXiv. https://doi.org/10.48550/arXiv.2101.00027.
[228] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶157.
[229] Gao, L., et al. (2020). The Pile: An 800GB Dataset of Diverse Text for Language Modeling (arXiv:2101.00027). arXiv. https://doi.org/10.48550/arXiv.2101.00027.
[230] About OpenStax. (n.d.). Kendall Hunt Publishing Company. Retrieved May 23, 2026, from https://he.kendallhunt.com/openstax.

[redacted]

322. The same reasoning applies at the level of individual books. As I explain above, no individual text or category of texts is necessary to an LLM's capability, and the utility of an LLM does not come from preserving the content of any subset or category of texts. I understand that Dr. Callison-Burch reaches a similar conclusion: the Asserted Works, if used, were a negligible fraction of the total training data and had no meaningful impact on model performance. Ultimately, no individual work or set of works is required for a diverse distribution of data, and the inclusion of any individual document should not have a meaningful impact on model outputs.

323. Dr. Zhao states that the technical value books provide for training LLMs is "closely tied" to their copyright status.[231] Copyright is a legal attribute, not a property of training data. If dataset preprocessing removes copyright management information (CMI), attribution, or other text associated with a work, that may change a given input token sequence supplied to the model during training. However, the technical consequence is the inclusion or exclusion of particular tokens. The model does not tokenize, embed, or update its parameters differently because the source work is copyrighted. If the same passage from a copyrighted book appeared in an openly licensed or public-domain source, the training signal would still only depend on the sequence of tokens and surrounding context, not on the work's copyright status. As I explain above, LLM training operates by converting text into tokens and representations, then adjusting model parameters to learn statistical relationships across the training data. The features identified in Dr. Zhao's analysis, such as contemporary language, editorial polish, genre diversity, and sustained narrative coherence, are textual attributes that may affect data quality, but they are not

[231] Hurst Decl. Ex. 251, Zhao Opening Report, ¶70. Dr. Zhao asserts that the characteristics he describes (contemporary language, editorial polish, genre diversity, sustained narrative coherence) are specific to commercially published, copyrighted works; that public-domain collections such as Project Gutenberg consist predominantly of works published before 1928, written in the language conventions of an earlier era; and that the technical value books provide for training modern LLMs is closely tied to their copyright status, i.e., that in his view it is contemporary copyrighted fiction, not books as an abstract category, that supplies the signal Defendants sought.

[redacted]

copyright-only attributes. Dr. Zhao's analysis conflates copyright status with technical value.

324. Testimony does not support Plaintiffs Experts' claim that model developers treated books, news, or any other "high quality" category as uniquely necessary to model capability. Researchers admitted that there was intuition that certain classes of data might be high quality by nature, but that "in practice, that has not really panned out."[232] Books were described as just one category among many sources of "diverse material" that have been considered during dataset development.[233]

B. News- and book-like text can improve related outputs but are not required for general LLM capability; Plaintiffs overstate the functional importance of news and books

325. Dr. Wenger[234] and Dr. Etzioni claim that LLMs are able to mimic the style and content of news articles. Dr. Zhao similarly highlights the value of books for long, coherent, linguistically rich text.[235] At most, these statements suggest that exposure to news-like or book-like text may improve a model's ability to generate news-like or book-like outputs. They do not show that news or books have unique functional importance to general LLM capability. In general, the distributional properties of LLM outputs are derived from the properties of the text that it is trained on. In order to be more generally useful, a model should be trained on a wide diversity of types of text.

---

[232] Hurst Decl. Ex. 16, Jan. 1, 2026 30(b)(1) Deposition of Greg Brockman at 77:23-78:5.

[233] Hurst Decl. Ex. 104 Jan. 29, 2026 30(b)(6) Deposition of Nicholas Ryder at 44:8-45:9; Hurst Decl. Ex. 111.A Sohl Deposition Ex. 420 (showing the wide variety of datasets GPT-4 was trained on).

[234] Hurst Decl. Ex. 246, Wenger Opening Report §II.A. Dr. Wenger claims that, at a high level, a pretrained generative model can mimic the types of content it is trained on; that a model trained on a significant number of news articles will be able to generate outputs in the style of news articles, while a model without news articles in its pretraining data likely would not have that capability or would not do so as effectively; and that to the extent OpenAI wanted to train a model to effectively deliver content like current events and analysis of important issues, the model would uniquely benefit from news articles in the training data.

[235] Hurst Decl. Ex. 251, Zhao Opening Report, ¶67; Simon Rowberry, The Value of Books in the Age of Generative AI Training Data, in Convergence: The International Journal of Research into New Media Technologies, 31(6), pages 1935-1950, 1945 (2025) (quoting Alec Radford, et al., Improving Language Understanding by Generative Pre-Training, OpenAI Preprint (2018)).

[redacted]

326. A model can learn from the factual information, current events, explanatory writing, multiple perspectives, and domain-specific language that can appear in many sources. No single alternative source would necessarily provide the same training value as a broad data mixture. The characteristics and attributes Plaintiffs' Experts identify are not unique to licensed news content or to any specific publisher's articles. As I explain in Section XI.A, similar attributes can appear in sources such as freely available news content, Wikipedia, public-domain materials, government publications, academic and technical documentation, blogs, tutorials, and other open-web text. The availability of those characteristics across many source types undermines the claim that licensed news content is technically necessary to achieve general LLM capability.

327. Dr. Etzioni claims that generative models can reproduce news style and content.[236] However, the ability to generate news-like style or content is what one would expect from a model that has learned recurring statistical patterns of a genre. Additionally, general news writing should be distinguished from a prompt that specifically asks the model to imitate the idiosyncratic style of a particular writer, columnist, or editorial voice. Such a prompt would be a targeted generation task conditioned on the prompt and could be a function of grounding on data from that writer instead of evidence that general news-like writing requires any specific publisher's content.

328. The linguistics literature describes news reports as a general written register, meaning a variety of language associated with recurring situational and linguistic characteristics, such as informational purpose, focus on current events, factual reporting separated from opinion pieces, and attribution to sources.[237] Lower-level linguistic features of news writing include frequent use of nouns, noun premodifiers, attributive adjectives, past-tense verbs, and time and place

[236] Hurst Decl. Ex. 250, Etzioni Opening Report, ¶123.
[237] Biber, D., & Conrad, S. (2012). Register, Genre, and Style. https://doi.org/10.1017/CBO9780511814358. pg. 111- 117, 125.

[redacted]

adverbials.[238] To the extent a model generates news-like writing, this is consistent with learning recurring register- or genre-level patterns that appear across many sources. It does not show that any specific publisher's content was uniquely necessary. As I explain above, LLMs can generalize because they learn across many subjects, writing styles, and forms of expression during training.

329. Dr. Zhao makes a similar but book-specific claim based on long contiguous prose, sustained narrative or argument, editorial polish, and genre diversity.[239] Those features may be useful training signals for discourse-related capabilities, but they are not unique to commercially published books. Public-domain books, including books available through Project Gutenberg, can also contain long contiguous prose, sustained narrative, and varied genres. More broadly, long-range coherence, exposition, narrative structure, and polished language are genre-level and discourse-level patterns that can be learned across many sources of extended or organized text. A model's ability to produce coherent long-form output does not show that any particular category of commercially published books, much less any individual book, is required for general LLM capability.

330. Writing style is not defined by a single word, phrase, sentence, or passage. It is recognizable from recurring patterns across many features of language, including word choice, sentence structure, use of pronouns and nouns, attribution, sequencing, cohesion, discourse organization, and conventional text structure. News reports, novels, essays, technical materials, and other forms of writing differ because those features tend to appear in different combinations and at

[238] Biber, D., & Conrad, S. (2012). Register, Genre, and Style. https://doi.org/10.1017/CBO9780511814358. pg. 118- 122.

[239] Hurst Decl. Ex. 251, Zhao Opening Report, ¶70. Simon Rowberry, The Value of Books in the Age of Generative AI Training Data, in Convergence: The International Journal of Research into New Media Technologies, 31(6), pages 1935-1950, 1945 (2025) (quoting Alec Radford, et al., Improving Language Understanding by Generative Pre-Training, OpenAI Preprint (2018)); Paul Keller, et al., Towards a Books Data Commons for AI Training, Technical Report, Creative Commons / Open Future (2024) (available at https://creativecommons.org/wpcontent/uploads/2024/04/240404Towards_a_Books_Data_Commons_for_AI_Training.pdf).

[redacted]

different rates across many examples.[240] A model exposed to a broad range of writing can learn these genre-level or register-level regularities as patterns of language, just as it can learn grammar, vocabulary, and factual associations from many examples.[241]

331. When an LLM does generate news-like or book-like text, it relies on generalization rather than dependence on any particular work. In other words, the ability to produce a type or style of writing is not the same as requiring any specific publisher's articles or any specific book. For example, a user can prompt an LLM, "Write a 500-word news article for a major news agency about a newly discovered planet in the solar system, including a headline, lead paragraph, background on observations, and quotations from a fictional astronomer,"

332. And it will produce a news-like article with attribution, factual sequencing, background context, and explanatory structure. If prompted to write a similar article in the style of "The Onion", it might produce an article with a humorous and satirical storyline, without reproducing any portion of a previously published piece.

333. Similarly, a user could prompt an LLM, "Create a twenty-chapter outline for an original fantasy sci-fi novel about a peasant in medieval Europe who suddenly obtains the ability to travel through time. Write the first chapter which introduces the setting and main character,"

334. And it will produce a coherent story with character consistency, foreshadowing, narrative escalation, and resolution. These examples reflect learned patterns about how different kinds of writing are structured. They do not show that the model depends on any particular publisher's articles, any commercially published books, or the expressive content of any specific work.

---

[240] Biber, D., & Conrad, S. (2012). Register, Genre, and Style . https://doi.org/10.1017/CBO9780511814358. pg. 9, 15-16.

[241] Biber, D. (1986). Spoken and Written Textual Dimensions in English: Resolving the Contradictory Findings. Language , 62 (2), 384–414. https://doi.org/10.2307/414678. pg. 385, 390-393, 397-400, 404-407.

[redacted]

335. No single work, and no particular subset of works, is necessary to train a capable LLM. Model capability derives from training across a large and diverse corpus as a whole; for any given set of works, including the works of any particular publisher, substitute text of similar kinds could confer the same general capabilities. Establishing rigorously that any particular work or set of works made a measurable difference to a trained model's capabilities would require an ablation study, meaning training matched models with and without the material at issue, and Plaintiffs' experts have conducted no such study. Indeed, it is impossible to ascribe responsibility for any particular model behavior to any one particular piece of training data.

XII. THE SMALL NUMBER OF PLAINTIFF-DOMAIN REFERENCES IN THE DEUCALION FINE-TUNING DATA DOES NOT ESTABLISH THAT PLAINTIFF CONTENT MADE A MEASURABLE CONTRIBUTION TO DEUCALION'S PERFORMANCE

336. Dr. Goldstein describes the fine-tuning of GPT-3.5 Turbo to create Deucalion using, among other OpenAI datasets, synthetic Bing Chat conversation data provided by Microsoft.[242] Dr. Goldstein found references to Plaintiff domains in two shards of that data. Specifically, Dr. Goldstein identifies 25 occurrences of nytimes.com in conversations_0901-1.jsonl and 20 occurrences of nytimes.com in conversations_0907-2.jsonl, resulting in 45 occurrences of nytimes.com in total, of which 5 are references to Wirecutter.[243]

337. The presence of URLs that are cited in the Deucalion synthetic fine-tuning data does not establish that they contributed to Deucalion's performance. In a similar manner to pre-training, RLHF fine-tuning adjusts a model's behavior through the aggregate effect of many training examples; the influence of any individual reference, or small set of references, is not established by its presence in the data. Dr. Goldstein identifies a count of references and characterizes Deucalion's

[242] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶181.
[243] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶181; Hurst Decl. Ex. 113, 30(b)(6) Deposition of Xia Song, January 13, 2026, p. 42; Hurst Decl. Ex. 199, MSFT_AI_MDL_002644093.

[redacted]

responses as improved, but he offers no analysis connecting the two. He does not isolate the effect of the cited URLs or the content of the web pages on Deucalion's behavior, and he does not establish that the model's improved responses depended on Plaintiff content rather than on the fine-tuning process itself. Establishing such a connection would require running an ablation study, training models with and without the questioned data; this has not been done.

338. These counts should be evaluated against the full size of the fine-tuning data used for Deucalion training. Without an appropriate denominator, 45 occurrences of nytimes.com (including the 5 Wirecutter references) do not establish that Plaintiff content made a measurable contribution to Deucalion's performance. In the same manner the synthetic conversations with Bing grounding data were made available to Plaintiffs' Experts, this data was made available to my team who ran an analysis to calculate the appropriate denominator of messages, which was found to be 228,679.

339. With this denominator, the number of references to nytimes.com and Wirecutter stand at a vanishingly small 45/228,679 or 0.02% of the synthetic conversations with Bing grounding data.[244] A contribution of that magnitude does not by itself support attributing Deucalion's improved performance to Plaintiff content. Microsoft's internal documentation attributes the purpose of this fine-tuning effort to improving [redacted] not to any particular publisher's content.[245] The characterization of Deucalion's responses as [redacted] is consistent with post-training designed to teach the model to use context more effectively and to produce richer grounded answers, and does not establish that Plaintiff content caused the improvement.

340. Post-training data, including the Deucalion synthetic fine-tuning data, can be used to improve instruction following, response richness, factuality, and grounded answer quality. These improvements, which allow Deucalion to generate [redacted] responses, are consistent with learning from

[244] 45 / 228,679 = 0.0001967824 * 100 = 0.02%.
[245] Hurst Decl. Ex. 199, MSFT_AI_MDL_002644093 at slide 43.

[redacted]

aggregate patterns across the post-training data, not with any specific publisher's content being responsible for the model's capabilities. References or snippets from the Deucalion fine-tuning data can help train the model to use grounding context more effectively, but their presence in post-training data does not mean the model memorized or stored Plaintiff articles and content as retrievable text in parameters.

XIII. COPILOT EXHIBITS VANISHINGLY LOW LEVELS OF OUTPUT REGURGITATION OF PLAINTIFFS' NEWS AND BOOKS CONTENT

341. Plaintiffs' three experts who attempt to analyze the regurgitation rates of Plaintiffs' Asserted Works by Copilot (Drs. Goldstein, Shan, and Wenger) have each adopted flawed assumptions and criteria which lead to inflated results. Dr. Goldstein's analysis in particular selects a short n-gram threshold, or matching value of 4 (i.e., the number of words which constitutes a match), without a principled methodological basis and utilizes different n-gram thresholds for different analyses without explanation. Even with these flawed assumptions, Plaintiffs' Experts regurgitation analyses still find very low rates of regurgitation in Copilot, ranging from 24 out of 8.2 million (0.0003%) conversations[246] to 8.5M out of 191M (4.5%) messages, depending on the methodology.[247]

342. Critically, all of Plaintiffs' Experts regurgitation analyses rely on a subset of Copilot conversations resulting from keyword and domain searches related to the Plaintiffs' Asserted Works (e.g., "nyt " and "nytimes.com") across the [redacted] and [redacted] fields in the Chat Log Data. This choice has two critical flaws: a. First, it is inappropriate to measure model regurgitation using the [redacted] field, as this field represents grounding data not shown to users. Only the [redacted] field contains model responses shown to users. b. Second, from the entire population of more than 1 billion Copilot conversations, the keyword and domain

[246] 24 / 8,195,308 = 0.000002929 * 100 = 0.0003%.

[247] 8,583,560 / 191,442,357 = 0.044836 * 100 = 4.5% Summing the "[redacted]Count" row in Dr. Goldstein's opening analysis table amounts to 8,583,560. 191,442,357 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains.

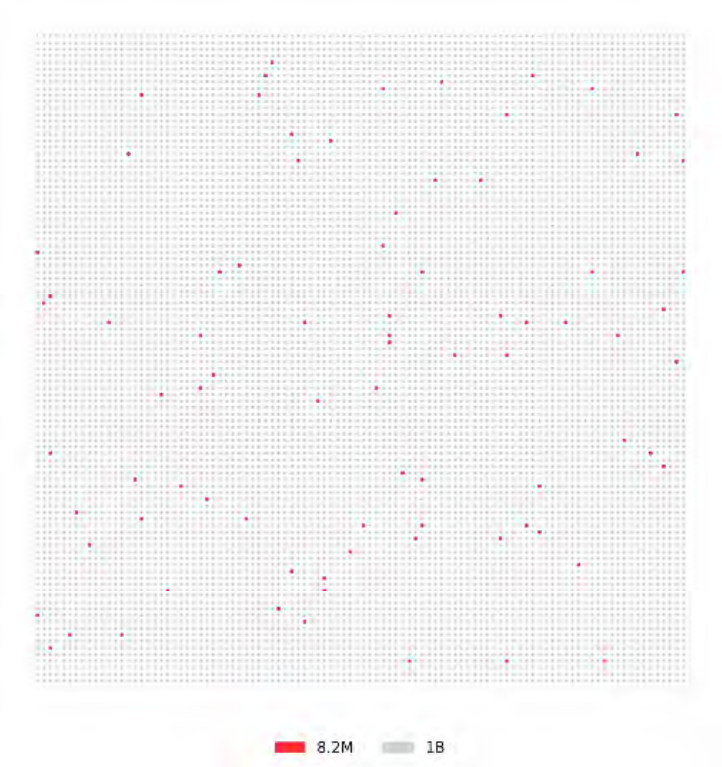
[redacted]

search for Plaintiff-affiliated terms resulted in approximately 8.2 million conversation hits ("Copilot conversations which contain the Plaintiff keyword or domains"), which represents just ~0.82% of the total population. Plaintiffs' Experts regurgitation analyses, which already report exceedingly low rates of regurgitation, rely on the 8.2 million figure as their denominator. However, because the set of Copilot conversations containing Plaintiff keyword or domains is drawn from searches related to Plaintiffs' Asserted Works, one would expect the likelihood of regurgitation to be higher among these conversations. In other words, the subset of conversations used for Plaintiffs Experts' analyses is biased. Table 2 below summarizes Plaintiffs' Experts regurgitation analyses directly from their reports and using the alternative 1 billion conversation denominator.

343. In Figure 1 below, I visualize the difference in size between the 1 billion conversation denominator and the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains. Critically, Plaintiffs' Experts solely rely on this biased subset of 8.2 million conversations and ignore the broader population of more than 1 billion total Copilot conversations.

Figure 1 (Rebuttal): Visualization of Difference Between Full Copilot Data and Subset

[redacted]



344. Even with these methodological errors, it is striking to note the already very low regurgitation rates presented by Drs. Wenger, Shan, and Goldstein, with Dr. Goldstein's results being comparatively higher given his non-conservative and flawed parameter choices that lead to inflated regurgitation rates, which I discuss further below. Dr. Goldstein finds 4.5% regurgitation, Dr. Wenger finds 0.0094% regurgitation (corrected to 0.0096% using the proper denominator), and Dr. Shan finds 0.0003% regurgitation. Without even correcting for the methodological choices in Plaintiffs' Experts analyses, and rather just using a denominator representing the entire population of Copilot conversations as opposed to the biased denominator of the Copilot conversations including Plaintiff keyword or domains, the percentages drop to 0.9% for Dr. Goldstein's analysis, 0.00008% for Dr. Wenger's analysis, and 0.000002% for Dr. Shan's analysis. These figures are summarized in Table 2 (Rebuttal) below.

[redacted]

Table 2 (Rebuttal): Plaintiffs Experts' N-Gram Regurgitation Results

#	Plaintiff Expert	Analysis Overview	Match Percent (8.2M dataset)[248]	Extrapolated Match Percent (1B conversations)[249]
1	Goldstein	Performs a 4-gram matching analysis over all News Plaintiffs works	4.5% ([redacted])[250] 11.4% ([redacted])[251]	0.9% ([redacted])[252] 1.1% ([redacted])[253]
2	Wenger[254]	Performs a 16-gram matching analysis over Mother Jones, The Intercept, and Reveal News works	0.0006% (For Mother Jones Registered Works)[255] 0.0096% (For Mother Jones, The Intercept, and Reveal News Asserted Works)[256]	0.000005% (For Mother Jones Registered Works)[257] 0.00008% (For Mother Jones, The Intercept, and Reveal News Asserted Works [258]

[248] The Shan and Wenger Analyses count the number of conversations while the Goldstein Analysis counts the number of messages.

[249] The Shan and Wenger Analyses count the number of conversations while the Goldstein Analysis counts the number of messages.

[250] 8,583,560 / 191,442,357 = 0.0448. 0.0448 * 100 = 4.5% (Where 191,442,357 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[251] 10,922,091 / 95,769,219 = 0.114. 0.114 * 100 = 11.4% (Where 95,769,219 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[252] 8,583,560 / 1,000,000,000 = 0.00858356. 0.00858356 * 100 = 0.9%. (Assuming 1B [redacted] messages).

[253] 10,922,091 / 1,000,000,000 = 0.010922091. 0.010922091 * 100 = 1.1%. (Assuming 1B [redacted] messages).

[254] Wenger approximates the Copilot Chat Logs to contain 8.4 million conversations, which is roughly 200,000 more conversations than were produced. I apply a denominator of 8.2 million in this table for consistency. Dr. Wenger described that she received a set of output logs from Microsoft Copilot including full logs of approximately 8.4 million conversations between users and chatbots, as well as internal log messages saved by Copilot as it processed user queries.

[255] Dr. Wenger calculated a conversation-level hit rate of 51/8,400,000, or 0.0006%, in the subset of Copilot outputs she inspected. Applying the corrected 8.2M denominator: 51 / 8,195,308 = 0.000006. 0.000006 * 100 = 0.0006%.

[256] Dr. Wenger's opening analysis estimates 280, 494, and 15 conversations present in the 8.2M Copilot conversations which contain the Plaintiff keywords or domains. 280 + 494 + 15 = 789. 789 / 8,195,308 = 0.000096. 0.000096 * 100 = 0.0096%.

[257] 51 / 1,000,000,000 = 0.000000051. 0.000000051 * 100 = 0.000005%.

[258] 789 / 1,000,000,000 = 0.000000789. 0.000000789 * 100 = 0.00008%.

[redacted]

3	Shan	Performs a 10-gram matching analysis over all Books Plaintiffs works	0.0003%[259]	0.000002%[260]

345. In the sections below, I detail and contrast Plaintiffs' Experts regurgitation analyses, describe the flawed assumptions with each design choice, and finally I will describe and show results that provide a more robust and conservative analysis that removes the critical design flaws found within these three reports.

A. Dr. Goldstein Relies on Flawed Assumptions Leading to Significantly Inflated Copilot Regurgitation Rates for News

346. Dr. Goldstein utilizes an n-gram matching approach to identify News Plaintiffs' material found within the Copilot output logs.[261] To accomplish this, Dr. Goldstein's analysis analysis searches for 4-grams across News Plaintiffs' material (e.g., "in the United States" counts as a match) and creates buckets of these matches based on hit occurrences of 6, 7, 8-20, 21-50, 51-70, 71-200, and 200+. Dr. Goldstein presents a table with the number of matches in each bucket (his "Results Table").[262] I note that Dr. Goldstein's Results Table presents inaccurate values in the "Total" column, which are incorrect by more than 25% in the case of the [redacted] counts.[263]

347. The analysis methodology in Dr. Goldstein makes several critically flawed design choices which lead to non-conservative estimates, an inflated number of matches, and false-positive examples.

---

[259] 24 / 8,195,308 = 0.0000029285. 0.0000029285 * 100 = 0.0003%.Dr. Shan, applying his methodology to Microsoft's output logs, identified reproductions of 30 or more near-verbatim words from 10 Plaintiff works across 24 conversations and 24 chat responses.

[260] 24 / 1,000,000,000 = 0.000000024. 0.000000024 * 100 = 0.000002%.

[261] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398. Dr. Goldstein ran an n-gram search over Microsoft's Copilot output logs using n=4, which he reports resulted in over a million hits having at least 20 matching 4-grams with News Plaintiffs' content and over a hundred thousand hits having at least 50 matching 4-grams.

[262] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398.

[263] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398. Summing the "[redacted] Count" and "[redacted] Count" rows in Dr. Goldstein's opening analysis table after ¶398 amounts to 8,583,560 and 10,922,091, respectively, compared to the reported 8,583,318 value for both [redacted] and [redacted].

[redacted]

348. First, Dr. Goldstein's Results Table match count percentage uses the total number of matches as its denominator for the per-bucket match percent, rather than using the total number of messages searched over. As a result, the percentage presented by Dr. Goldstein does not represent a regurgitation rate over the population of data searched. Instead, it merely shows the percentage that each per-bucket hit count represents of the total number of hits. In other words, Dr. Goldstein's percentages sum to 100% across his own buckets; they describe the distribution of hits among buckets, not the share of messages containing a hit. I discuss this in more detail below.

349. Based on my review of the produced source code, it is my understanding that Dr. Goldstein's Results Table counts the number of [redacted] and [redacted] messages.[264] Thus, the "Total" values in the rightmost column of Dr. Goldstein's Results Table are a sum of messages that have had at least 6 4-gram hits. Critically, the total value is not the total population of messages over which Dr. Goldstein conducted his own analysis. Specifically, there are 191 million messages in the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains which Dr. Goldstein's analysis searched over. The "Total" values that Dr. Goldstein's Results Table presents are just the total number of messages hits, not the total number of messages searched over, which is misleading as Dr. Goldstein presents the hit count within each bucket as a percent of the "Total".

350. Thus, using the total number of messages in the [redacted] field as a denominator to the sum of the [redacted] row of Dr. Goldstein's Results Table results in a total of 4.5% of messages containing at least 6 4-grams.[265] Similarly, using the total number of messages in the [redacted] field as a denominator to the sum of the [redacted]

---

[264] The produced code (ArticleLoader.java and Searcher.java) process the data at the message level based on its core architectural design. The system reads input files line-by-line, treating each line as a single message that is independently scored through the n-gram matching algorithm. Each message that exceeds the match threshold generates one output row containing that message's unique identifier and individual match statistics, confirming that messages are the fundamental unit of analysis.

[265] 8,583,569 / 191,442,357 = 0.0448. 0.0448 * 100 = 4.5%. (Where 191,442,357 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains.).

[redacted]

row of Dr. Goldstein's Results Table results in a total of 11.4% of messages containing at least 6 4-grams.[266]

351. Second, Dr. Goldstein does not filter the data's [redacted] field for "bot", thus capturing both user prompts and model outputs in the data. This is inappropriate as Dr. Goldstein attempts to conduct an output regurgitation analysis, but also captures user prompts.

352. Third, the analysis selects 4-grams as the unit for which a sequence of text is considered a match, which is not a conservative approach. The analysis searches each conversation in the Copilot conversations containing Plaintiff keywords or domains for just 4-grams (roughly speaking, about 4 words, constructed after the implementing source code normalizes the text by lowercasing it and stripping all digits and punctuation) that match any sequence of words across all News Plaintiff material. For example, Goldstein counts 4-grams such as "in addition to the", "at the same time", and "in the bay area" as matches.[267] Rather, a sensitivity analysis using multiple n-gram numbers, such as 8-gram, 12-gram, 16, and so on, should be used to assess the threshold at which regurgitation actually occurs. It is also striking that Plaintiffs' other experts, Wenger and Shan, who conduct the same regurgitation analysis, adopt more conservative n-gram values of 16 and 10 respectively.[268][269]

353. To further illustrate the scale of what a 4-gram match looks like in context, I show that the 4-gram matching criteria in contrast to the size of an individual work is miniscule.

Table 3 (Rebuttal): Fraction of 4-word Gram per Asserted Work

Copyright Owner	4-word Gram Fraction of Asserted Work
Chicago Tribune Company	0.64%

[266] 10,922,091 / 95,769,219 = 0.114. 0.114 * 100 = 11.4%.
[267] Hurst Decl. Ex. 231, Goldstein Opening Report Exhibit 18 at p. 16, 37.
[268] Hurst Decl. Ex. 246, Wenger Opening Report, pp. 105-110.
[269] Hurst Decl. Ex. 241, Shan Opening Report, ¶38.

[redacted]

San Jose Mercury-News	0.66%
The New York Times	0.67%
DP Media Network	0.73%
Northwest Publications	0.74%
Orb Publishing	0.75%
Daily News	0.84%
Sun-Sentinel Company	0.89%
Orlando Sentinel Communications Company	0.90%

354. Fourth, the analysis in Dr. Goldstein searches across all News articles within a given Copilot conversation, which means many disparate fragments of text across unrelated News articles are being matched to a single conversation, as opposed to using a per-article matching methodology. For example, if there are two separate News articles, one containing "in the Bay Area" and the second containing "at the same time", Dr. Goldstein's methodology would count 2 regurgitated matches against the following conversation text, despite that these sentence fragments are not contiguous in the original News articles: "I had lunch in the Bay Area while texting at the same time".

355. Fifth, the analysis in Dr. Goldstein's analysis use a windowing technique which counts overlapping n-grams within a sequence of text multiple times. For example, it would take the phrase "the quick brown fox" and, if creating 4-grams, would make two distinct 4-grams: "the quick brown fox" and "quick brown fox jumps". Dr. Goldstein then goes on to show that if a message contains "the quick brown fox jumps", this counts as 2 4-gram matches, despite the overlap between the matches being considerable. For the biggest column of 200+ 4-gram matches, this means that the conversation text could be as few as 204 words long, and could be assembled from fragments across multiple unrelated articles. The combination of these two factors can lead to severely over-inflated numbers for counting matches.

356. Sixth, the analysis in Dr. Goldstein searches across both internal logs ([redacted]) and output logs ([redacted]) shown to the user. Including the internal logs does not actually measure output regurgitation because [redacted] reflects

[redacted]

the grounding context the system assembles to form a response, not text presented to the user. Only the [redacted] field reflects user-facing output, which is what a regurgitation analysis should measure. To the extent the [redacted] figures are offered to show copying in the grounding process rather than in the output, the methodological choices described above (the 4-gram threshold, the absence of a contiguity requirement, and many-to-one matching) apply to those figures as well, so they do not establish that a substantial portion of any article was reproduced in the grounding data.

357. Last and most critically as described above, Dr. Goldstein's analysis operates on the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains, which was not randomly sampled from the larger universe of more than 1 billion Copilot conversations, but was rather selected by filtering based on a series of Plaintiff-associated keyword searches and URL matches, including keywords such as "New York Times", which were agreed upon by the parties. Above, I use the total number of [redacted] and [redacted] messages in the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains to present a denominator for the regurgitation study. I also calculate an extrapolated match rate using a denominator based on the more than 1 billion conversations in the Copilot logs.

358. Dr. Goldstein's "Selected Examples of Copilot Reproducing Asserted Works" exhibits, of which Exhibit 21 (The New York Times) is representative, do not show what they are offered to show. Each exhibit reproduces a series of conversation logs with spans of text highlighted in red, described as "spans of text that appear in the Asserted Works." Examined against the produced conversation logs and the code that generated the exhibits, the highlighting and the accompanying "16-gram score" do not establish that Copilot's user-facing outputs regularly reproduced substantial verbatim portions of the Asserted Works. There are three core reasons why: a. First, the inclusion criterion includes matches on grounding content rather than user-facing input.. The code admits a conversation if it shares a single 16-gram with any candidate article, which can be seen among the matches listed in Exhibit 21. The reported "16-gram score" is a count of

[redacted]

shared sixteen-word sequences computed across the entire conversation, much of which is retrieved grounding content rather than user-facing output. For example, Example 5 in Exhibit 21 has a 16-gram score of 210, but the 16-gram matches are all contained in the "internalsearchresulthidden" field, with the user-facing chats only containing a handful of 4-grams, including "in the united states," "to forgive up to," "less than … a year," "who received pell grants," "for an additional … in" and "of federal student loan debt for." Across the produced conversation files I analyzed, the [redacted] fields alone hold several times the character volume of the user-facing chat responses, and together with the text of the internal search-query and search-result messages the grounding content is several times the volume of the user-facing output. The high scores that some examples report are a function of that grounding content: where the retrieval system has fetched the full text of an article into the grounding fields, the conversation will share a large number of 16-grams with that article regardless of what the model output to the user. The score therefore measures overlap with the retrieved-and-pooled text, not the quantity of article text in the response. b. Second, the highlighting overstates what was matched, in two compounding ways. Although the score is computed on a 16-gram basis, the red highlighting is applied at the 4-gram level: the generation code builds a set of four-word sequences from the candidate articles and marks any four-word sequence in the conversation that appears in that set. The highlighted spans therefore appear longer and more continuous than the underlying four-word matches that generated them. c. Third, the user-facing output reproduces little of the grounding content. In the produced conversations I analyzed, only a small amount of the distinct 16-grams appearing in the bot's user-facing chat responses also appear in the same conversation's grounding fields; the user-facing output predominantly summarizes or rewords the retrieved material rather than reproducing it.

359. Taken together, these exhibits neither accurately indicate output regurgitation nor data used in grounding because they rely on the same flaws in Dr. Goldstein's

[redacted]

broader output-log analysis and none of these exhibits establishes that the model stored or reproduced a substantial portion of any Asserted Work.

B. Dr. Wenger Relies on Flawed Assumptions Leading to Inflated Regurgitation Rates in Dr. Wenger's Copilot Regurgitation Analysis

360. Dr. Wenger performs an n-gram analysis similar to that of Dr. Goldstein's analysis for the Mother Jones, Reveal News, and The Intercept Asserted Works. Dr. Wenger conducts this analysis by performing a 16-gram matching algorithm on the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains before filtering hits to those which had at least five 16-grams matching only one of Plaintiffs' articles. For reference, a 16-gram match would be only 2.8% of the text of a work.

361. This algorithm is first run on a subset of Mother Jones Registered Works where the resulting hits are manually filtered to ensure the results do not include public domain documents, such as the Constitution or the Bible. These filtered hits are used as a numerator over the larger set of pre-filtered hits to calculate a "hit rate." This hit rate is calculated by Wenger as 51/8,400,000 or 0.00061%.[270] Correcting this denominator to use the actual 8.2 million Copilot conversations which contain the Plaintiff keywords or domains results in a hit rate of 51/8,195,308 or 0.00062%.

362. Then Dr. Wenger performs the same 5 16-gram matching methodology for the larger set of Asserted Works, but opts not to perform a manual review, claiming that it would be unduly burdensome due to the large number of hits. The total number of hits is below 10,000, which is roughly four and a half times smaller than the number of conversations that Professor Bail and his team manually reviewed for Dr. Bail's analysis. Dr. Wenger's analysis instead multiplies the resulting matches by the previously computed Mother Jones hit rate to estimate the number of "true hits". In this stage of Dr. Wenger's analysis analysis, the total

[270] Hurst Decl. Ex. 246, Wenger Opening Report, p. 106. Dr. Wenger calculated a conversation-level hit rate of 51/8,400,000, or 0.0006%, in the subset of Copilot outputs she inspected.

[redacted]

hit rate is calculated as 789/8,400,000 or 0.0094%.[271] Correcting this denominator to use the actual 8.2 million Copilot conversations which contain the Plaintiff keywords or domains results in a nearly identical hit rate of 789/8,195,308 or 0.0096%.

363. Dr. Wenger's analysis proceeds in two stages, and its figures should be understood separately. In the first stage, Dr. Wenger analyzes a subset of Mother Jones Registered Works and reports 51 conversations containing at least five 16-gram matches, out of 8,400,000 conversations — a conversation-level hit rate of 0.0006%. Applying an assumed 7.5 million average daily Copilot users, one conversation per user per day, and 365 days, Dr. Wenger's analysis annualizes this to 45 conversations per day, or 16,425 conversations per year, with at least five 16-gram hits in Mother Jones Registered Works. In the second stage, Dr. Wenger estimates per-publisher figures by applying an assumed "true hit rate" derived from keyword filtering (8% at the five-hit threshold) to the pre-filtering hit counts for each publication. Using the corrected denominator of 8,195,308 Copilot conversations, this yields an estimated 494 conversations for Mother Jones (0.006%; 160,939 per year), 280 for The Intercept (0.0034%; 91,432 per year), and 15 for Reveal News (0.0002%; 4,954 per year). The 789 figure on which Dr. Wenger relies is the sum of these three estimates (494 + 280 + 15), and the corresponding annual figures sum to 257,325 conversations per year across the three publications.[272]

364. First, Dr. Wenger's first-stage hit rates of 0.0006% and 0.0096% are already exceedingly low. Even the second-stage per-publisher estimates, which Dr. Wenger's analysis inflates through the assumptions described below, remain a small fraction of one percent of conversations (0.0002%–0.006%). However, each stage rests on a distinct flawed assumption. The first stage extrapolates from

[271] Hurst Decl. Ex. 246, Wenger Opening Report, p. 106. Dr. Wenger calculated a conversation-level hit rate of 51/8,400,000, or 0.0006%, in the subset of Copilot outputs she inspected. Dr. Wenger's opening analysis estimates 280, 494, and 15 conversations present in the 8.2M Copilot conversations which contain the Plaintiff keywords or domains. 280 + 494 + 15 = 789.

[272] Hurst Decl. Ex. 246, Wenger Opening Report, p. 106, 109.

[redacted]

the 8.2 million keyword- and domain-filtered subset as though it were representative of the full population of Copilot conversations. These Copilot conversations contain Plaintiff keyword or domains and are not randomly sampled from the larger population of Copilot conversations, but were rather selected by filtering based on a series of Plaintiff-associated keyword searches and URL matches, including keywords such as “Mother Jones”, which were agreed upon by the parties. Dr. Wenger’s assumption that the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains is representative of the larger population of Copilot conversations is therefore flawed, and as a result so is Dr. Wenger’s extrapolation logic. The second stage compounds this with a further assumption: it applies an 8% “true hit rate,” derived from keyword filtering on Mother Jones Registered Works, uniformly to The Intercept and Reveal News, treating a rate validated on one publication's works as applicable to two others without support.. Using the 1 billion conversations as the denominator, the first-stage Mother Jones figure of 51 conversations corresponds to a rate of 0.000005%, and the second-stage aggregate of 789 conversations corresponds to 0.00008%. Either way, measured against the population Dr. Wenger purports to describe, the rate is several orders of magnitude below the figures the report emphasizes.

365. Further, Dr. Wenger’s analysis assumption that the 789 hits are distributed evenly across time instead of clustered around certain times and events (e.g. a particularly popular article was published) is similarly flawed, and this becomes an issue when Wenger extrapolates the number of conversations using the average daily Copilot users. The 8.2 million Copilot conversations which contain the Plaintiff keyword or domains include information pertaining to the dates of each conversation, yet Dr. Wenger contains no analysis on the distribution of these conversations over time to determine if there is such a temporal pattern.

[redacted]

C. Dr. Shan Relies on Flawed Assumptions Leading to Inflated Copilot Regurgitation Rates for Books

366. First, I understand from counsel that Microsoft is not accused of book-related output regurgitation in the same way addressed here; I nevertheless respond to Dr. Shan's opening analysis for completeness.[273] Dr. Shan conducts this analysis using a 10-gram search with a sliding window, while considering a match if a sequence has at least 30 words.[274] The resulting number of conversations that were flagged by this analysis is 24, or 0.0003% of all conversations in the 8.2 million Copilot conversations, which contain the Plaintiff keyword or domains, within the time period.[275] To demonstrate the scale of three 10-gram matches in comparison to the text that is being matched against, I use the length of each of the Asserted Works in the Books data provided by Dr. Shan's analysis. Three 10-gram matches would be only 0.024% of the entirety of the work, a very small amount. I note that the raw word counts (totaling 26,347,043 words), derived from the versions of the Asserted Works relied upon in Dr. Shan's analysis, are overstated because they contain substantial boilerplate material, back matter, and recurring page elements. Further, several of the files reflect entire anthologies or compilations rather than the specific short fiction or editorial contributions that constitute the asserted works in those files. As a result, this word count is a conservative value.

367. Dr. Shan's analysis involves matching on all messages generated by Copilot, including those shown to the user ([redacted]) and messages that are internally used for grounding ([redacted]). This approach does not actually measure regurgitation, as the [redacted] field represents grounding information that is not shown to the user and is used to ground responses. It is normal during the grounding process for snippets of material to be used in these internal messages

[273] *In Re: OpenAI, Inc. Copyright Infringement Litigation*, 1:25-md-03143, (S.D.N.Y. Jun 13, 2025) ECF No. 183.
[274] Hurst Decl. Ex. 241, Shan Opening Report, ¶156.
[275] 24 / 8,195,308 = 0.0000029. 0.0000029 * 100 = 0.0003%.

[redacted]

to generate a response. Only messages that are actually presented to the user ([redacted]) should be counted for the presence of regurgitated material.

368. Similar to the other experts and as I describe above, Dr. Shan's analysis operates on the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains, which was not randomly sampled from the larger population of Copilot conversations, but was rather selected by filtering based on a series of keyword searches and URL matches, including keywords such as "New York Times", which were agreed upon by the parties. The analysis in the Shan report for books is performed on the dataset that was created by filtering for News Plaintiffs' keywords and domains. Dr. Shan's assumption that the 8.2 million Copilot conversations which contain the Plaintiff keyword or domains is representative of the larger population of more than 1 billion Copilot conversations is flawed. Extrapolating to the denominator representing the population of the more than 1 billion conversations, assuming no further matches, results in a 0.000002% hit rate.[276]

D. Analysis Correcting the Errors Present Across Plaintiffs Experts' Copilot Regurgitation Analyses

369. To address the critical flaws in Plaintiffs' Experts output matching analyses and, specifically, Dr. Goldstein's significantly inflated results, I first recreate Dr. Goldstein's analysis using the configurations and design decisions to the best of my understanding given the deficiencies in their productions. Specifically, I use the following configurations from Dr. Goldstein's analysis: a. N-gram value of 4-grams b. Matches are many-to-one (where many disparate articles' n-grams can count towards the bucketed counts on a message)[277] c. Overlapping n-grams can be counted multiple times against a message d. Does not filter [redacted] for "bot", thus capturing both user prompts and model outputs

[276] 24 / 1,000,000,000 = 0.000000024. 0.000000024 * 100 = 0.000002%.
[277] For example, if one article had "in the United States" and another article had "it was at that" and both 4-grams were found in the message, this would count as 2 distinct matches.

[redacted]

370. The results of my replication of Dr. Goldstein's analysis output matching analysis are below. While the per-bucket values do not match Dr. Goldstein's results exactly,[278] the total values match exactly when Dr. Goldstein's bucket counts are summed.[279] For the [redacted] field, I identify 8,583,560 hits and Dr. Goldstein identifies 8,583,560 hits. For the [redacted] field, I identify 10,922,091 hits and Dr. Goldstein identifies 10,922,091 hits. This provides confidence that our analyses share the same foundational logic, but that my results are more appropriate given the conservative methodological choices.

371. Correcting for the flawed methodological design decisions in Dr. Goldstein's analysis, I use the following parameters: a. N-gram value of 16-grams[280] b. Matches are one-to-one (where only one article's n-grams can count towards the bucketed counts for a message) c. Overlapping n-grams are not counted multiple times d. [redacted] is "bot" (to filter out user messages and thus only capture model outputs)

372. I adopt each parameter to address a specific defect identified above: the 16-gram threshold reduces matches on common short word sequences that do not reflect reproduction of protectable expression; one-to-one attribution prevents a single message from being counted as multiple matches assembled from fragments of unrelated articles; and non-overlapping counting prevents a single span of text from being counted multiple times. I do not contend that any single parameter setting identifies the threshold at which reproduction occurs; I report results across a more appropriate range of parameter settings to obtain matches to Plaintiffs' asserted works.

373. Additionally, I segment the results based upon model responses that are shown to the user ([redacted]) and grounding data that is not shown to the user ([redacted])

---

[278] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398.

[279] Summing the "[redacted] Count" and "[redacted] Count" rows in Dr. Goldstein's opening analysis table amounts to 8,583,560 and 10,922,091, respectively, compared to the reported 8,583,318 value for both [redacted] and [redacted].

[280] I chose 16-gram to mirror the selection by Dr. Wenger's analysis, which similarly performs a regurgitation analysis. Additionally, a 16-gram is appropriate for this exercise as it skips over common phrases, factual fragments, and headlines.

[redacted]

and limit the messages to only those with [redacted] value "bot" to represent model output.

374. In Table 4 below, I compare my results directly in contrast to Goldstein to emphasize the egregious inflation of numbers within their methodology. While Dr. Wenger's approach demonstrates inefficiencies within their calculation and extrapolations, the errors are much smaller in comparison to Goldstein.

375. In total, I find 35,806 messages with at least 6 16-gram matches to News articles. That represents ~0.42%[281] of the hits identified by Dr. Goldstein's analysis, underscoring how Dr. Goldstein's flawed design decisions lead to severely inflated results. Similarly, I find that there are 88,178 matches within the [redacted] field (i.e., the grounding data), which represents just ~0.81% of the hits identified within the Goldstein report.[282]

376. Further, the 35,806 matches are just 0.02% of all non-empty "bot" messages in the biased 8.2 million Copilot conversations which contain the Plaintiff keyword or domains. Similarly, the 88,178 matches within the [redacted] field are just 0.09% of all the non-empty [redacted] messages in the biased 8.2 million dataset.

377. Combining the two fields, the share of messages containing Plaintiff content under the corrected parameters, whether in the user-facing output ([redacted]) or the grounding field ([redacted]), is between 0.05% for the non-empty messages in the 8.2 million Copilot conversation subset[283] and 0.006% of all messages in the 1 billion conversations, which I conservatively assume to also contain only 1 billion messages for both [redacted] and [redacted] (i.e., just one message per conversation)

[281] 35,806 / 8,583,560 = 0.00417146265. 0.00417146265* 100 = 0.42%.
[282] 88,178 / 10,922,091 = 0.0080733625 * 100 = 0.81%.
[283] (35,806 + 88,178) / (158,047,588 + 95,719,689) = 0.0004885. 0.0004885 * 100 = 0.05%.

[redacted]

Table 4 (Rebuttal): News Output Matching Analysis Results

Expert (Data Field)	Messages Matched	Match Percent (8.2M dataset)[284]	Extrapolated Match Percent (1B conversations)[285]
Goldstein[286] ([redacted])	8,583,560	4.5%[287]	0.9%[288]
Lafferty ([redacted])	**35,806**	**0.02%[289]**	**0.004%[290]**
Goldstein ([redacted])	10,922,091	11.4%[291]	1.1%[292]
Lafferty ([redacted])	**88,178**	**0.09%[293]**	**0.009%[294]**

378. Similarly, I also perform an equivalent 16-gram matching exercise over the Books data produced by Dr. Shan's analysis and find that the match percent is lower that reported, at 4 conversations for [redacted] (0.000003% of conversations) and 8 conversations for [redacted] (0.000008% of conversations).

[284] 191,442,357 total non-empty [redacted] messages present in the 8.2M Copilot conversations, which contain the Plaintiff keyword and domains. 158,047,588 total non-empty "bot" [redacted] messages in the 8.2M copilot conversations, which contain the Plaintiff keyword and domains. The Goldstein and Lafferty analyses count the number of messages, while the Wenger analysis counts the number of conversations.

[285] The Goldstein and Lafferty analyses count the number of messages, which I conservatively assume to be equal to the number of conversations at 1 billion.

[286] *See* Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398.

[287] 8,583,560 / 191,442,357 = 0.0448. 0.0448 * 100 = 4.5% (Where 191,442,357 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[288] 8,583,560 / 1,000,000,000 = 0.009. 0.009 * 100 = 0.9% (Assuming 1 message per conversation).

[289] 35,806 / 158,047,588 = 0.00022. 0.00022 * 100 = 0.02% (158,047,588 total non-empty "bot" [redacted] messages in the 8.2M copilot conversations, which contain the Plaintiff keyword and domains.).

[290] 35,806 / 1,000,000,000 = 0.0000358. 0.0000358 * 100 = 0.004%.

[291] 10,922,091 / 95,769,219 = 0.114. 0.114 * 100 = 11.4% (Where 95,769,219 is the number of non-empty [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[292] 10,922,091 / 1,000,000,000 = 0.0109. 0.0109* 100 = 1.1%.

[293] 88,178 / 95,719,689 = 0.0009. 0.0009 * 100 = 0.09% (Where 95,719,689 is the number of non-empty "bot" [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).

[294] 88,178 / 1,000,000,000 = 0.000088. 0.000088 * 100 = 0.009% (Assuming 1B [redacted] messages).

[redacted]

379. It should be noted, however, that the 8.2 million dataset is a biased dataset that was constructed from agreed-upon keywords by the Parties, so these results are only to correct the methodology found within Dr. Shan's analysis. Relatedly, if we assume there are no additional matches to the Books data within the population dataset of more than 1 billion conversations, which I also conservatively assume contain 1 billion [redacted] and [redacted] messages (i.e., one message per conversation), then an extrapolated percent of matched messages would be 0.0000004% for [redacted] and 0.0000008% for [redacted]

Table 5 (Rebuttal): Books Output Matching Analysis Results

Expert (Data Field)	Messages Matched	Match Percent (8.2M dataset)[295]	Extrapolated Match Percent (1B conversations)[296]
Shan[297] ([redacted] and [redacted])	24	0.0003%[298]	0.000002%[299]
Lafferty ([redacted])	4	0.000003%[300]	0.0000004%[301]
Lafferty ([redacted])	8	0.000008%[302]	0.0000008%[303]

380. Before evaluating what Plaintiffs' experts' hit counts show, I replicated their mechanical counts and confirmed that I could reproduce them from the produced

[295] Dr. Shan's analysis counts the number of conversations while my analysis counts the number of messages.
[296] Dr. Shan's analysis counts the number of conversations while my analysis counts the number of messages; I assume the number of messages is equal to the number of conversations at 1 billion.
[297] Hurst Decl. Ex. 241, Shan Opening Report, ¶156.
[298] 24 / 8,195,308 = 0.0000029285. 0.0000029285 * 100 = 0.0003%. Dr. Shan, applying his methodology to Microsoft's output logs, identified reproductions of 30 or more near-verbatim words from 10 Plaintiff works across 24 conversations and 24 chat responses.
[299] 24 / 1,000,000,000 = 0.000000024. 0.000000024 * 100 = 0.000002%.
[300] 4 / 158,047,588 = 0.0000000253. 0.0000000253 * 100 = 0.000003% (Where 158,047,588 is the number of non-empty "bot" [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).
[301] 4 / 1,000,000,000 = 0.000000004. 0.000000004 * 100 = 0.0000004% (Assuming 1B [redacted] messages).
[302] 8 / 95,719,689 = 0.0000000835. 0.0000000835 * 100 = 0.000008% (Where 95,719,689 is the number of non-empty "bot" [redacted] messages present in the 8.2M Copilot conversations which contain the Plaintiff keyword and domains).
[303] 8 / 1,000,000,000 = 0.000000008. 0.000000008 * 100 = 0.0000008% (Assuming 1B [redacted] messages).

[redacted]

data and code. The corrected results I present above are consistent across my analyses and, once appropriate denominators are applied, with the results of Plaintiffs' experts' own analyses.

E. Technical Details of My Rebuttal Analysis to Correct the Collective Errors Present Across Plaintiffs Experts' Analyses

381. This section describes the methodology employed in my rebuttal n-gram analysis. It evaluates textual overlap between source articles and target messages using an n-gram comparison methodology. An n-gram is a fixed-length sequence of consecutive words. For each source article and target conversation, the analysis identifies unique n-grams so that repeated phrases within the same document are not counted multiple times.

382. Specifically, the analysis uses 16-grams, meaning each n-gram consists of exactly 16 consecutive words. The choice of 16-gram length balances specificity and recall: longer sequences increase the likelihood that matches represent substantive textual overlap rather than common phrases, while still capturing meaningful similarity.

383. The analysis applies an author-based filter during the staging process to ensure that only bot-generated messages are included in the target dataset for n-gram comparison. This is because the bot-generated messages are the model outputs and are most appropriate to be evaluated for regurgitation. Specifically, the pipeline filters to messages where the value of the [redacted] field is "bot", excluding all user-generated messages from the analysis.

384. Before n-gram generation, all text undergoes normalization to ensure consistent matching. The normalization process performs the following transformations in sequence: (1) replaces underscores and hyphens with spaces, (2) removes all non-letter, non-space characters (keeping only [a-zA-Z\s]), (3) converts all text to lowercase, (4) splits the text on whitespace into a word array, and (5) filters out empty strings. This standardization ensures that variations in punctuation, capitalization, and formatting do not prevent legitimate textual matches from being identified.

[redacted]

385. The analysis compares the unique n-grams from the source materials against those appearing in the target messages to identify overlapping language. When the same n-gram appears in both a source article and a target conversation, it is recorded as a match. The methodology accounts for situations in which the same language appears in multiple source articles or in multiple messages within a conversation, while preserving the attribution of each match to the relevant source and target materials.

386. After matches are identified, the results are deduplicated so that each unique match between a particular conversation, article, and n-gram is counted only once. The final results are aggregated at the individual message level. The resulting scores reflect the number of unique overlapping n-grams between each source-target pair. Deduplication is performed using groupBy operations on (message_id, article_id, ngram_text) tuples.

387. The results are then converted into message-level summaries. For book matches, the analysis calculates the number of unique messages associated with each book, distinguishing between matches found in visible text and matches found in hidden text fields. For article matches, the analysis produces comparable aggregate statistics across the article dataset, including unique message counts and total hit counts for both visible and hidden text which are presented in the tables above.

XIV. NEWS URLS ARE RETRIEVED BY COPILOT'S GROUNDING SYSTEM INFREQUENTLY; THE GOLDSTEIN REBUTTAL REPORT'S NEWS-RELATED GROUNDING ANALYSIS IS NOT APPROPRIATE TO MEASURE THE RELATIVE PRESENCE OF NEWS AND PINK SLIME DOMAINS

388. The Times's expert Professor Emily Bell prepared lists of news and alleged pink slime-related domains (the "Bell News List" and "Bell Pink Slime List," respectively) for use in Dr. Goldstein's analysis of news and alleged pink slime

[redacted]

material present in the Copilot log data.[304] The Bell News List is compiled from Palewire's homepages.news, the Medill STATE of Local News Project 2025, and the World Media Directory, and is substantially broader than the Plaintiffs' Asserted Works domain list.[305] Specifically, domains associated with Plaintiffs' Asserted Works represent just 0.5% of the entire Bell News List. As a result, Dr. Goldstein analyzes a broad list of news-related domains as opposed to a list limited to domains potentially containing Plaintiffs' Asserted Works. For the Bell Pink Slime List, I do not offer an independent opinion on the overall credibility or reliability of Dr. Bell's classification. I note, however, that the list includes domains, such as geeksforgeeks.com and investing.com, that do not appear to be ordinary examples of AI-generated pink slime sites.[306]

389. As described above, I prepared a random sample of 45,000 Copilot conversations reflecting general consumer Copilot usage. Dr. Goldstein analyzes that sample by counting conversations in which the grounding fields include at least one domain from the Bell News List or Bell Pink Slime List. Of the 45,000 conversations, Dr. Goldstein states that 33,259 conversations contained data in the grounding fields, and that 5,967 of those conversations contained a Bell News List hit and 842 contained a Bell Pink Slime List hit.[307] Those counts are presented as 17.94% and 2.53% of Copilot conversations with grounding data (i.e., [redacted] field), respectively.

390. However, this conversation-level analysis is answering the wrong question, as it does not measure the relative prevalence of News, alleged Pink Slime, Plaintiff-

---

[304] Hurst Decl. Ex. 229, Bell Rebuttal Report, ¶29; Hurst Decl. Ex. 229, Bell Rebuttal Report, ¶¶206-211; Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶109-113.

[305] Hurst Decl. Ex. 229, Bell Rebuttal Report, ¶207.

[306] MDL Dkt. 398, Appendix A; NewsGuard: Global Leader in Information Reliability, available at https://www.newsguardtech.com/aboutnewsguard/ (last accessed May 27, 2026); Professor Bell compiles the News list from Palewire's homepages.news (created by a Reuters journalist), the Medill State of Local News Project, and the US-based publishers from the World Media Directory; Professor Bell compiles the Pink Slime list from a NewsGuard tracker produced from a NYT subpoena, a list of pink slime networks from the Tow Center for Digital Journalism, and a list of sites identified by the parties in this litigation relating to pink slime, Appendix A; NewsGuard: Global Leader in Information Reliability, available at https://www.newsguardtech.com/aboutnewsguard/ (last accessed May 27, 2026).

[307] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶113.

[redacted]

domain, or Asserted Work URLs among the total population of sources retrieved by Copilot's grounding system. Dr. Goldstein counts an entire conversation as a hit if at least one domain in the Bell News List or Bell Pink Slime List appears in the [redacted] field, even though a single conversation can include many grounding URLs unrelated to those lists.[308] Specifically, as shown in Table 1, the vast majority of URLs found within grounding data are not associated with the URLs or domains associated with Plaintiffs' Asserted Works or even news generally.

391. To correct for Dr. Goldstein's inflated statistics, I analyze the total URLs appearing in the [redacted] field and report the share of all grounding URLs. This measures the actual prevalence of news and pink slime domains found in the grounding field, as opposed to coarsely counting conversations as hits whenever any listed domain appears, as Dr. Goldstein does.[309] In addition to applying the generic Bell News List and Bell Pink Slime List, I also conduct this analysis using only the Plaintiff-operated domains.

392. Using this methodology, I find that Bell News List hits account for 10,417 URLs (4.20%, or about 1/24, of all grounding URLs), Bell Pink Slime List hits account for 1,001 URLs (0.40%, or about 1/250, of all grounding URLs), Plaintiff-only Domain hits account for 183 URLs (0.07%, or about 1/1,429, of all grounding URLs), and News Plaintiff Asserted Work hits account for 10 URLs (0.004%, or about 1/24,000, of all grounding URLs) in the 45,000 Copilot sample. By comparison, Dr. Goldstein's analysis finds Bell's News List domains hit on 17.94% of grounded conversations and Bell's Pink Slime List domains hit on 2.53% of grounded conversations.[310]

393. Table 1 (Reply) below compares Dr. Goldstein's conversation-level counts with my URL-level results.

---

[308] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶112-113.
[309] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶113.
[310] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶113.

[redacted]

Table 1 (Reply): Grounding Data URL Citation Analysis Results[311]

Category	Goldstein - Grounding Conversation Counts[312]	Goldstein - % Conversations with Grounding	Lafferty - Grounding URL Counts	Lafferty - URLs as % of all Grounding URLs
Bell News List	5,967 / 33,259[313]	17.94%	**10,417[314] / 248,049[315]**	**4.19%**
Bell Pink Slime List	842 / 33,259	2.53%	**1,001[316] / 248,049**	**0.40%**
News Plaintiff Domains	-	-	**183 / 248,049**	**0.07%**
News Plaintiff Asserted Works	-	-	**10 / 248,049**	**0.004%**

394. Critically, the Bell News List is broader than the News Plaintiffs domains and significantly broader than the Asserted Works. A domain list that includes non-Plaintiff news sources (e.g., The Guardian, Time, MSN, BBC, Fortune, etc.) cannot be used as a proxy for the presence of Plaintiffs' content. When the analysis is limited to Plaintiff-only domains, the count falls to 183 URLs out of 248,049 grounding URLs, or 0.07% of the total. Further, when the analysis is limited to Plaintiff Asserted works, the count falls to 10 URLs out of 248,049 grounding URLs, or 0.004%. This is consistent with my grounding analysis of the 8.2 million conversation Copilot dataset, where I found that only 1.42% of Plaintiff Asserted Work URLs appear in the grounding data, representing only 0.18% of all unique URLs used for grounding. Thus, even in a dataset selected

[311] 13 / 2,479 = 0.005244. 0.005244* 100 = 0.5%; Asserted Domains not in the Plaintiff Exhibit include nyt.com; nytcooking.com; cookingnytimes.com; cookingnyt.com; nytimescooking.com; cooking.nytimes.com; newyorktimescooking.com; cookingnewyorktimes.com; wirecutter.com.

[312] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶113, Table 8.

[313] Dr. Goldstein reports that 33,259 conversations included retrieval-augmented generation fields, including the [redacted] field in [redacted].

[314] Domains from Bell's News List appear 10,417 times in the grounding data.

[315] Total number of non-unique URLs in 45K Copilot sample.

[316] Domains from Bell's Pink Slime List appear 10,417 times in the grounding data.

[redacted]

using Plaintiff-related keywords and domains, Plaintiffs' Asserted Work URLs make up a very small share of the grounding sources. Dr. Goldstein's broader news domain conversation count does not alter that conclusion because it uses a different numerator and an inappropriate denominator that does not answer the appropriate question.

395. Regarding the "pink slime" analysis, Dr. Goldstein's conversation count similarly does not show what share of Copilot's retrieved URLs came from the Bell Pink Slime domains, nor whether the listed domains account for a meaningful share of the grounding context.[317] Using the URL-level denominator, I find that Bell Pink Slime List hits account for 1,001 total URLs (0.40% of all grounding URLs in the data). In addition, my review of the Bell Pink Slime List identified several domains that are questionably classified as pink slime, including domains that appear to be mainstream or otherwise reputable, such as geeksforgeeks.com and investing.com. Given this, I call into question the methodology with which this list was compiled, but choose to adopt it in my analysis for ease of comparison. Because I take the Bell Pink Slime List as given, my conclusions do not depend on any view of how the list was compiled: even accepting the list in full, the domains on it account for a fraction of one percent of the grounding URLs.

XV. REGURGITATION OF PLAINTIFFS' WORKS BY COPILOT IS EXCEEDINGLY RARE; SHARED TEXT BETWEEN GROUNDING DATA AND OUTPUTS DOES NOT ESTABLISH REGURGITATION

396. Dr. Goldstein's rebuttal analysis measures the extent to which there is overlapping text (16-grams) between the text in the grounding field and model outputs and counts as a "leak" any 16-gram match between the grounding material and the output, irrespective of whether the matched text involves Plaintiffs' content or an Asserted Work.[318] Thus, Dr. Goldstein's analysis does not quantify the extent to which the model's outputs (with or without grounding)

[317] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶113.

[318] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶77-82.

[redacted]

actually contain news or Plaintiffs' works. Additionally, the term "leak" itself is not an agreed-upon or commonly used term within the field of LLMs and grounding data, and the usage of this term misrepresents how the model works in normal operation, which can include cited references to grounded material. In other words, a "leak" from the grounding field, as Dr. Goldstein defines it, may be a natural property of grounding and not an example of regurgitation.

397. Above, I perform an analysis that addresses this issue by calculating the frequency of Plaintiffs' Asserted Work URLs in the grounding field of the 8.2 million subset of Copilot conversations containing Plaintiff keyword or domains, finding that only 1.42% of Asserted Work URLs are present in the grounding fields, which represents only 0.18% of all unique URLs in the grounding field. In other words, Plaintiff domains represent a very small proportion of Copilot grounding sources.

398. Still, considering the question Dr. Goldstein's rebuttal analysis asks regarding the extent to which there is shared text between the model output and grounding fields, the definition of a "leak" as regurgitation is fundamentally flawed. Dr. Goldstein treats the presence of the same 16-gram in both the grounding fields and the Copilot output as an example of regurgitation.[319] This description is a misnomer for how grounding technology works and its purpose. Properly understood, the presence of the same 16-gram in the grounding field and the output does not establish regurgitation. As I describe above, a key feature of grounding involves the separation between retrieval and text generation, where retrieval determines what content should be provided to the model to consider during text generation to the user. This allows the model to generate new token sequences from both its internal parameters and the grounded material without altering the model's parameters.[320] Some overlap between retrieved text and output, particularly of short factual spans, is an expected byproduct of summarizing retrieved material in natural language. As I further explain above,

[319] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶77-82.
[320] Supra Section IV.D.1.

[redacted]

the alignment process is used to shape grounded generation so that the model summarizes, synthesizes, and cites retrieved material rather than reproducing it verbatim, producing original responses that draw on, rather than copy, that material. Verbatim reproduction of a retrieved passage is the exception these measures are designed to limit, occurring in particular where a user expressly requests an exact quotation or specific information.

399. Dr. Goldstein analyzes shared text between the grounding data and the output field in the 8.2M subset of Copilot conversations which contain the Plaintiff keyword or domains. Dr. Goldstein performs this by filtering to conversations with Plaintiff domains in the grounding field and then searching for 16-grams (words) that appear both in the grounding field and in the user-facing output. Dr. Goldstein counts any 16-gram match between the grounding field and the user text output as shared text.[321] So although Dr. Goldstein's rebuttal analysis filters to conversations whose grounding contains a Plaintiff domain, it then counts a 16-gram match between the output and any text grounded upon in that conversation –– it does not examine whether the matched text is a Plaintiff work or derived from a Plaintiff domain. My review of Dr. Goldstein's source code corroborates that the content in the shared text is not necessarily content from Plaintiff articles, but could even be sourced from text generated by the model or even content from other non-Plaintiff sources retrieved by the model. Dr. Goldstein finds a total of 59,545 conversations with such shared text, or 0.73% of all conversations in the 8.2 million subset of Copilot conversations containing Plaintiff keyword or domains.[322] As I state above, this dataset is a biased sub-population for regurgitation-related analyses.

400. Dr. Goldstein's analysis filters to Plaintiff domains, but not necessarily Asserted Works.[323] I perform a corresponding shared text analysis filtering for Asserted

[321] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶77-79.

[322] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶82; 41,487+3,523+3,369+3,123+2,359+1,887+1,235+1,173+1,069+320 = 59,545. 59,545/8,195,308 = 0.00726. 0.00726 * 100 = 0.73.

[323] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶79, 82.

[redacted]

Works in the 8.2 million subset of Copilot conversations containing Plaintiff keywords or domains. To maintain consistency with the matching analysis above, I first apply the same 16-gram threshold; I then repeat the analysis at more demanding 25-gram and 40-gram thresholds. Assuming an average of six characters per word, a 25-gram threshold corresponds to roughly 150 characters and a 40-gram threshold corresponds to roughly 250 characters. Reporting at all three thresholds serves two analytical purposes. First, it measures the sensitivity of the 16-gram results to the choice of threshold. Second, it measures the length of the verbatim overlap itself: because a higher threshold counts a conversation only where the output and the grounding field share a longer unbroken run of identical words, comparing the counts across thresholds shows the extent to which the shared text consists of short matching sequences rather than longer ones.

401. I note that in my analysis, similar to Dr. Goldstein's rebuttal analysis, I do not perform additional verification to ensure the n-grams are sourced from the news articles themselves as opposed to other third-party sources referenced in the grounding field.[324] These results should accordingly be considered overinclusive, as the measured number of conversations with shared text between the grounding data and the output field could include instances where an asserted work URL was included in the grounding data but the shared text above the specified n-gram threshold was matched from a grounding source that was not actually an Asserted Work. I provide the results of my analysis in Table 2 (Reply) below. The results show, for each Plaintiff and n-gram threshold, the number of conversations in the 8.2 million subset in which the grounding fields include Plaintiff's Asserted Work URLs and the output shares an n-gram of at least the stated length with any text appearing in the grounding field. As with Dr. Goldstein's analysis, the shared text can be any content appearing in the

[324] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶77-82.

[redacted]

grounding field; it need not be an Asserted Work, and the analysis does not test whether it is.

Table 2 (Reply): Asserted Works Grounding Data Shared Text Analysis at Different N-Gram Thresholds

N-gram Threshold (Plaintiff)	Number of Shared Text Conversations	% of 8.2M Conversations[325]
16-gram (NYT)	3,147	0.038%[326]
16-gram (Daily News)	98	0.0012%
16-gram (Denver Post)	705	0.0086%
16-gram (Mercury News)	599	0.0073%
16-gram (Chicago Tribune)	363	0.0044%
16-gram (Twin Cities)	175	0.0021%
16-gram (Orlando Sentinel)	96	0.0012%
16-gram (Sun Sentinel)	55	0.00067%
16-gram (CIR)	46	0.00056%
16-gram TOTAL	**5,284**	**0.064%**
25-gram (NYT)	1,255	0.015%
25-gram (Daily News)	27	0.00033%
25-gram (Denver Post)	279	0.0034%
25-gram (Mercury News)	232	0.0028%
25-gram (Chicago Tribune)	93	0.0011%
25-gram (Twin Cities)	52	0.00064%
25-gram (Orlando Sentinel)	26	0.00032%
25-gram (Sun Sentinel)	13	0.00016%
25-gram (CIR)	15	0.00018%
25-gram TOTAL	**1,922**	**0.024%**
40-gram (NYT)	410	0.005%
40-gram (Daily News)	8	0.000098%
40-gram (Denver Post)	60	0.00073%
40-gram (Mercury News)	57	0.0007%
40-gram (Chicago Tribune)	24	0.00029%
40-gram (Twin Cities)	10	0.00012%
40-gram (Orlando Sentinel)	7	0.000085%
40-gram (Sun Sentinel)	1	0.000012%
40-gram (CIR)	5	0.000061%
40-gram TOTAL	**582**	**0.007%**

[325] Exact number of conversations is 8,195,308.
[326] 5,237 / 8,195,308 = ~0.0639%.

[redacted]

402. In Table 3 (Reply) below, I further segment the above 16-gram matches based on the publisher of the asserted works URLs found in the grounding data and find significantly lower hits than Dr. Goldstein's rebuttal analysis following the methodological correction.[327]

Table 3 (Reply): Grounding Data Shared Text Analysis Results Segmented by Publisher

Publisher	Goldstein - Conversations with Shared Text in Output and Grounding Data, Filtered for Plaintiff Domains	Lafferty - Conversations with Shared Text in Output and Grounding Data, Filtered for Asserted Works[328]
The New York Times	41,487	**3,147**
Denver Post	3,123	**705**
Mercury News	3,523	**599**
Chicago Tribune	3,369	**363**
Pioneer Press	1,173	**175**
New York Daily News	1,887	**98**
Orlando Sentinel	1,069	**96**
Sun Sentinel	1,235	**55**
Mother Jones (CIR)	N/A	**46**
Orange County Register	2,359	**0**
The Athletic	320	**0**

A. Dr. Goldstein's Supplemental Analyses Result in Exceedingly Low Hit Rates for Plaintiffs' Asserted Works

403. In his supplemental analysis, Dr. Goldstein conducts an analysis for the presence of Asserted Works in the Copilot output and grounding log fields.[329] A similar n-gram analysis was conducted in Dr. Goldstein's opening analysis;[330] however, this new supplemental analysis takes a more conservative approach that agrees more closely with my figures in Section XIII above and the analysis results of Plaintiffs' other Experts.

---

[327] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶82.
[328] Conversations where there exists an Asserted Work URL and at least 16-gram shared text between [redacted] and [redacted] fields.
[329] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶22-38.
[330] Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398.

[redacted]

404. To account for the privacy program redactions, in his supplemental analysis, Dr. Goldstein implements a "continuity score".[331] This continuity score searches for 16-gram sequences comprised of exact 4-gram matches allowing for a single word gap between each matching 4-gram that does not match the Asserted Works. This is a reasonable approach to account for the presence of redactions. This is one of many approaches that can be implemented to work around the privacy program redactions and demonstrates that the redactions do not materially impact the ability to analyze the output logs.

405. In his supplemental analysis, Dr. Goldstein describes an originality condition that screens the hits to ensure that text from the Asserted Work is not exclusively in the form of a quotation or recitation of an outside source.[332] The sole instrument enforcing this condition is the Gemini-3.1-Pro-Preview model, prompted to classify whether a span is the author's prose or quoted from elsewhere. In his supplemental analysis, Dr. Goldstein provides the system prompt, but does not provide any robustness or statistical basis for the results of the prompt.[333] The prompt instructs the model that "[i]f the highlighted text is a mix of author-written prose and third-party content, output 'yes'," meaning mixed spans are retained as matches. Because news articles can be quotation-dense, a substantial share of any matched span in a news article could be mixed text, and the prompt is set to count every mixed case as a hit. That is a bias in the screening step which leads to an overcount in the number of hits.

406. Dr. Goldstein's supplemental revised approach provides only the absolute counts for the number of Asserted Works that purportedly match grounding data and Copilot responses in the Copilot logs.[334] When these absolute counts are contextualized using the total number of Asserted Works as a denominator, the

[331] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶24-25; Hurst Decl. Ex. 231, Goldstein Opening Report, ¶387. Dr. Goldstein asserts that the redactions are likely to have had a negative impact on News Plaintiffs' ability to inspect and search the output logs and that the results of News Plaintiffs' searches over the approximately 8.2 million logs are underinclusive.
[332] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶27.
[333] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶28.
[334] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶36-38.

[redacted]

resulting percentages are shown to be exceedingly small. Specifically, the Copilot logs contain snippets of text that purportedly match only 0.1% of all Asserted Works, as shown in the tables below. My analysis in Section XIII above also finds very low regurgitation rates (0.02% for [redacted] and 0.09% for [redacted]), which agrees with Dr. Goldstein's supplemental analysis.[335]

407. For simplicity, given that Dr. Goldstein presents five slices of results in his supplemental analysis, the tables below only present corrections to the lowest thresholds of both the continuity score and 16-gram overlap score used by Dr. Goldstein's supplemental analysis.[336] These figures represent the largest values of hits computed by Dr. Goldstein's supplemental analysis and cover the entire range of possible matches in the data. Even with these generous thresholds, less than 0.1% of all Asserted Works contain a snippet in the Copilot logs.

408. Table 1 (Sur-reply) below presents the matching results from Dr. Goldstein's supplemental Tables 1 and 4[337] showcasing the number of purported Asserted Works matches in the Copilot grounding field using the total number of Asserted Works for each Plaintiff as denominators. This results in exceptionally low percentages.

[335] *Supra* Table 1 (Rebuttal).
[336] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶23-25, 28, 37; A continuity score of ≥16 and 16-gram overlap score of ≥1.
[337] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶37.

[redacted]

Table 1 (Sur-reply): Normalized Rates of Presence of Asserted Works in Copilot Grounding Fields

Plaintiff	Unique Asserted Works	Goldstein Asserted Works in Grounding (Continuity Score)	% of all Asserted Works (Continuity Score)	Goldstein Asserted Works in Grounding (16-gram)	% of all Asserted Works (16-gram)
New York Times	4,519,337	2,601	**0.058%**	2,462	**0.054%**
Chicago Tribune	1,205,350	1,200	**0.100%**	1,159	**0.096%**
San Jose Mercury News	130,331	931	**0.714%**	877	**0.673%**
DP Media Network	89,269	1,005	**1.126%**	969	**1.085%**
Daily News	525,341	356	**0.068%**	333	**0.063%**
Northwest Publications	47,527	326	**0.686%**	300	**0.631%**
Orlando Sentinel	424,822	289	**0.068%**	276	**0.065%**
Sun-Sentinel	134,518	121	**0.090%**	117	**0.087%**
Orb Publishing	35,603	36	**0.101%**	32	**0.090%**
TOTAL	7,112,098	6,865	**0.097%**	6,525	**0.092%**

409. Table 2 (Sur-reply) below presents the matching results from Dr. Goldstein's supplemental Tables 2 and 5[338] showcasing the number of purported Asserted Works matches in the Copilot output field (i.e. Copilot responses) to users using the total number of Asserted Works for each Plaintiff as denominators. This results in exceptionally low percentages.

[338] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶37.

[redacted]

Table 2 (Sur-reply): Normalized Rates of Presence for Asserted Works in Copilot Outputs to Users

Plaintiff	Unique Asserted Works	Goldstein Asserted Works in Output (Continuity Score)	% of all Asserted Works (Continuity Score)	Goldstein Asserted Works in Output (16-gram)	% of all Asserted Works (16-gram)
New York Times	4,519,337	183	**0.0040%**	104	**0.0023%**
Chicago Tribune	1,205,350	82	**0.0068%**	45	**0.0037%**
San Jose Mercury News	130,331	94	**0.0721%**	57	**0.0437%**
DP Media Network	89,269	123	**0.1378%**	78	**0.0874%**
Daily News	525,341	15	**0.0029%**	6	**0.0011%**
Northwest Publications	47,527	21	**0.0442%**	12	**0.0252%**
Orlando Sentinel	424,822	17	**0.0040%**	9	**0.0021%**
Sun-Sentinel	134,518	9	**0.0067%**	5	**0.0037%**
Orb Publishing	35,603	1	**0.0028%**	1	**0.0028%**
TOTAL	7,112,098	545	**0.0077%**	317	**0.0045%**

410. The backup materials produced with Dr. Goldstein's supplemental analysis identify, for each matched Asserted Work, the specific Copilot conversations containing the purported match.[339] Counting the unique conversations across those lists, a purported match appears in the Copilot output field in only 604 conversations under the continuity score and 332 conversations under the 16-gram score, out of the 8,195,308 conversations in the 8.2 million subset: a

[339] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶37.

[redacted]

conversation-level rate of 0.0074% and 0.0041%, respectively.[340] These conversation-level rates, derived from Dr. Goldstein's own supplemental analysis, are consistent with, and lower than, the corrected rates I calculate in Section XIII (0.02% for [redacted] and 0.09% for [redacted]) and the shared-text rate presented in this Section (0.064%).

B. Example Outputs Presented by Dr. Goldstein's supplemental analysis Are Not Examples of Regurgitation

411. As discussed above, text from Plaintiffs' works appearing in a grounding field is not regurgitation. Rather, it reflects the ordinary operation of the grounding process and does not by itself establish regurgitation. In his supplemental analysis, Dr. Goldstein describes 5 examples from the 45,000 conversation sample, where in each example, text from an Asserted Work appears in a grounding field.[341] Since a grounding field contains material returned by the retrieval system, it is expected that a retrieved snippet may include verbatim text from its source.

412. These examples merely describe the normal grounding process, as discussed above. Grounding combines a pre-trained language model with an informational retrieval system that supplies relevant material as temporary context for a response. The retrieved material enters the inference context for that response and does not alter, retrain, or update the model.

413. As discussed above, there is a distinction between the retention of facts and statical patterns, memorization of expression from a training example, and reproduction of that expression in a user-facing output. Regurgitation refers to the reproduction of expression from a training example in a user-facing output. [redacted] in a grounding field is temporary retrieval context and its presence does not establish that the model reproduced text memorized from its training data.

---

[340] 604 / 8,195,308 = 0.0000737. 0.0000737 * 100 = 0.0074%. 332 / 8,195,308 = 0.0000405. 0.0000405 * 100 = 0.0041%. Extrapolated to 1 billion conversations: 604 / 1,000,000,000 * 100 = 0.00006%; 332 / 1,000,000,000 * 100 = 0.00003%.

[341] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶55-60.

[redacted]

414. Further, the presence of shared text in grounding and output fields does not by itself establish regurgitation, as explained above. As I also explain below, the selected examples in Dr. Goldstein's supplemental analysis relied heavily on grounding content, and the user-facing output predominantly generated new text conveying information contained in the retrieved material.[342] The examples in Dr. Goldstein's supplemental analysis likewise show that the retrieval system supplied material as temporary inference-time context for the model while largely generating new text to the user. They do not by themselves establish regurgitation of the Asserted Works.

XVI. PLAINTIFFS' EXPERTS' CLAIMS ON LLM MEMORIZATION AND COMPRESSION REST ON THEORETICAL LITERATURE THAT DOES NOT APPLY TO MODERN LLMS

415. The opinions in this Section respond to the challenges by Dr. Wenger and Dr. Zhao to my discussions above of generalization and of model weights, respectively. They support my opinions (offered above) that LLM training produces models that generalize across the training distribution rather than store the training data, and that regurgitation of training text is a real but bounded phenomenon. The Plaintiffs' experts' theoretical claims do not change those opinions, and in any event the frequency with which regurgitation actually occurs is an empirical question that I address through direct measurement, not theory.

A. Dr. Wenger's Claim that Generalization Requires Memorization Relies on a Flawed Read of the Cited Literature

416. Dr. Wenger claims that it is established and well-explained that generalization requires memorization, supported primarily by a citation to a 2021 paper by Gavin Brown et al. and two papers by Vitaly Feldman published in 2020 and

[342] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶55-60.

[redacted]

2021.[343] In making this claim, Dr. Wenger responds to my discussion above of generalization, where I explain that the objective of LLM training is to improve the model's ability to respond to new and unseen prompts by learning statistical patterns across the training distribution, rather than by memorizing specific training examples.[344] Dr. Wenger's categorical claim that generalization requires memorization does not hold for language modeling, for the reasons I describe below. Dr. Wenger's claim rests on theoretical models appearing in the Brown and Feldman papers, which are highly questionable when applied to language modeling as I describe further below.

417. The Brown and Feldman papers rely on theoretical analysis that has little relevance to language modeling. I am familiar with this style of information-theoretic analysis, having used similar methods in my own work, and with the conditions under which its results hold.[345] As is common in theoretical computer science research, these analyses adopt simplified assumptions in order to state precise mathematical results; those assumptions suit the formal models the papers study, but they do not apply more broadly to real-world modern language models. The conclusion Dr. Wenger's rebuttal analysis draws about LLM training is one the papers themselves do not support.[346]

418. I first discuss the Brown paper, noting two fundamental problems with using this analysis to argue that "generalization requires memorization" in LLMs. First, the conclusion in the paper is based on average mutual information, which is a statistical quantity similar to correlation. As a result, and as the authors of the

[343] Hurst Decl. Ex. 248, Wenger Rebuttal Report, pp.8-10; Brown, G., et al. (2021). When is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning? Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing , 123–132. https://doi.org/10.1145/3406325.3451131; Feldman, V., & Zhang, C. (2020). What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation (arXiv:2008.03703). arXiv. https://doi.org/10.48550/arXiv.2008.03703; Feldman, V. (2021). Does Learning Require Memorization? A Short Tale about a Long Tail (arXiv:1906.05271). arXiv. https://doi.org/10.48550/arXiv.1906.05271.

[344] Hurst Decl. Ex. 248, Wenger Rebuttal Report, p.75.

[345] See, Zhou, S., et al. (2009). Compressed and Privacy-Sensitive Sparse Regression. IEEE Transactions on Information Theory , 55 (2), 846–866. https://doi.org/10.1109/TIT.2008.2009605.

[346] Hurst Decl. Ex. 248, Wenger Rebuttal Report, pp.8-10.

[redacted]

paper themselves acknowledge, the theorem in the Brown paper does not directly address the question of memorization, since it does not show that the training data can be efficiently extracted from the prediction algorithm.[347]

419. Second, and just as important, the independent sampling scheme used in the Brown paper is an over-simplification of how data are collected and used for training LLMs. In particular, the constructions are set up so that "subpopulations" are independent, with many "singleton" examples. Real language data, as used in training LLMs, is very different from this sampling scheme. A local context corresponds to a subpopulation, and the contexts have significant statistical dependencies. Words, phrases, and topics co-occur and share structure in ways that allow a learning algorithm to generalize across subpopulations, which is precisely what the independence assumption of this paper rules out. But this use of statistical dependence across contexts is how LLMs generalize. The authors acknowledge that independence between subpopulations is a significant and unrealistic simplification for models of natural language.[348]

420. Moving onto the Feldman papers, these papers are mostly concerned with tasks like image or document labeling/classification, not language modeling. In image classification, an image is input and the learning algorithm must classify it ("It's a picture of dog"). In document classification, the learning algorithm just categorizes its content ("It's an article about doping in cycling."). The Feldman papers' use of long-tailed distributions are based on the empirical observation that the distribution of the label categories tends to follow a heavy-tailed (or long-tailed) Zipf-like distribution.

---

[347] Brown, G., et al. (2021). When is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning? Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing , 123–132. https://doi.org/10.1145/3406325.3451131. ("Our results lower bound mutual information. The statements do not directly shed light on whether a computationally efficient attacker, given access to the classifier, could recover some or all of the training data.").

[348] *Id.*

[redacted]

421. Similar to the Brown et al. paper discussed above, the Feldman papers use independent sampling of subpopulations, which is invalid for language modeling. Language modeling makes sequential predictions, one token at a time, in highly overlapping contexts that have strong statistical dependence.

422. The nature of the language modeling problem is also qualitatively different. In LLMs, the model may regurgitate or generate parts of the training data. This is different from the label memorization of categories that may have a heavy-tailed distribution. Memorization in LLMs is more likely to occur when text is often repeated in the training data, which is the opposite of the types of rare categories in long-tailed distributions or outliers that are analyzed in these papers.

423. In summary, Dr. Wenger's categorical statement that "generalization requires memorization" is unjustified and misleading when applied to training of LLMs.[349] Dr. Wenger makes this broad claim by resting on highly stylized theoretical models and assumptions in order to make mathematical statements, and these assumptions and the resulting mathematical conclusions are not directly valid for LLMs.[350]

B. Dr. Zhao's Claim that LLM Training is Equivalent to Compression Misrepresents the Cited Literature

424. Dr. Zhao argues that training a language model is equivalent to compressing the training data, with or without loss.[351] This analogy responds to my discussion above of model weights, which explains that training does not create a repository of the texts used to train the model, but instead produces numerical parameters that encode statistical relationships derived from the training corpus. Dr. Zhao is

[349] Hurst Decl. Ex. 248, Wenger Rebuttal Report, p.75.
[350] Hurst Decl. Ex. 248, Wenger Rebuttal Report, pp.7-12.
[351] Hurst Decl. Ex. 252, Zhao Rebuttal Report, ¶¶14, 23-41, 54; David, O., et al. (2016). On statistical learning via the lens of compression. Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016). https://dl.acm.org/doi/pdf/10.5555/3157382.3157410; Delétang, G., et al. (2024). Language Modeling is Compression. 12th International Conference on Learning Representations (ICLR 2024). https://openreview.net/pdf?id=jznbgiynus; Conklin, H. C., et al. (2026). Learning is Forgetting: LLM Training As Lossy Compression (arXiv:2604.07569). arXiv. https://doi.org/10.48550/arXiv.2604.07569.

[redacted]

incorrect to equate training LLMs with compression in two fundamental ways. First, if a model can predict well, it can compress well, but the converse is not generally true. Second, the theoretical literature that relates prediction to compression is based on idealized assumptions that simply do not hold for LLMs, as I explain below.

425. Elaborating on my first point, information theory shows that prediction is primary. The major classical results in information theory use probabilistic models of the source.[352] Using arithmetic coding or other coding algorithms, in principle an LLM could be used to compress new data (or its training data) up to the cross-entropy of the model with respect to the held-out data.[353] This cross-entropy is equivalent to the validation loss used to assess generalization. In the reverse direction, a compression algorithm like Lempel-Ziv (the basis for standard modern compression tools such as zip), which may do a good job of compressing a given data set, cannot be readily turned into a statistical model that assigns accurate probabilities to tokens in new data as LLMs do.

426. Elaborating on my second point, from a theoretical perspective, the literature relating compression to prediction is based on idealized mathematical assumptions, in particular that the source being modeled is stationary and ergodic.

427. A source is stationary if its statistical properties do not change over time.[354] For a stationary source, the probability of seeing a particular sequence of tokens does not depend on when in the sequence you look at it. An ergodic source is a stronger condition. In addition to being stationary, an ergodic source has the property that its time averages equal its ensemble averages. Informally, this means that a single, sufficiently long sample from the ergodic source is

[352] Shannon, C. E. (1948). A Mathematical Theory of Communication. The Bell System Technical Journal. https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.
[353] Witten, I. H., et al. (1987). Arithmetic coding for data compression. Communications of the ACM, 30 (6), 520–540. https://doi.org/10.1145/214762.214771.
[354] Specifically, it means that the joint probability distribution of any sequence of symbols is invariant under time shifts.

[redacted]

representative of the whole source; the source doesn't get "stuck" in sub-patterns or regimes. In the literature relating prediction to compression, this is referred to as the Shannon-McMillan-Breiman theorem.[355] By contrast, natural language, which is the data LLMs are used to predict, is neither stationary nor ergodic.

428. Natural language is clearly not stationary. Topics shift, vocabulary drifts across decades, style varies by genre, and long-range dependencies are pervasive. The distribution over words is non-uniform across time in obvious ways. Natural language is even less plausibly ergodic. Different documents, authors, genres, and eras constitute very different "regimes." A single book, no matter how long, will never statistically resemble the full distribution of all human language. At best, AI researchers wishing to draw on ideas from information theory may treat the data as approximately locally stationary and ergodic, over short spans or within a fixed domain or genre. In a 1951 paper by the father of information theory, Claude Shannon, the entropy of the English language was estimated by treating it as an ergodic source, but Shannon acknowledged that this was an idealization.[356]

429. As a concrete example, consider the use of the word "like" in American English, which has several distinct uses. In particular, the quotative "like", for example as used in the sentence "She was, like, 'no way'", emerged in California youth speech in the 1980s, and is now widespread. Similarly, the word is now commonly used as a filler, as in "it was, like, really strange".[357]

430. As a result, the 6-gram "it was, like, really" will have very different statistics over time. Moreover, observing the 6-gram will shift the entire conditional distribution going forward, to reflect this more informal word choice. This basic

---

[355] *See* Breiman, L. (1957). The Individual Ergodic Theorem of Information Theory. The Annals of Mathematical Statistics, 28(3), 809–811. https://doi.org/10.1214/aoms/1177706899; Cover, T. M., & Thomas, J. A. (2006). Elements of Information Theory (2nd Edition). Wiley-Interscience. https://www.wiley.com/en-us/elements-of-information-theory-2nd-edition-p-9780471241959.

[356] Shannon, C. E. (1950). Prediction and Entropy of Printed English. The Bell System Technical Journal. https://www.princeton.edu/~wbialek/rome/refs/shannon_51.pdf.

[357] Romaine, S., & Lange, D. (1991). The Use of like as a Marker of Reported Speech and Thought: A Case of Grammaticalization in Progress. American Speech, 66 (3), 227–279. https://doi.org/10.2307/455799; Vandelanotte, L., & Davidse, K. (2009). The emergence and structure of be like and related quotatives: A constructional account. Cognitive Linguistics, 20 (4). https://doi.org/10.1515/COGL.2009.032.

[redacted]

fact points to how theoretical equivalences between compression and prediction are invalid in practice.

431. In summary, it is at best misleading to equate language modeling with compression. Prediction is primary, and an algorithm that compresses the training data well may not be readily turned into an accurate model of prediction for new data. Furthermore, the theoretical literature in information theory that relates prediction to compression is based on strong mathematical assumptions that are patently invalid for human language of the form that is used as training and validation data for LLMs.

XVII. CONCLUSION

432. The discussion above shows that modern LLMs are the product of decades of development in AI, computer science, and natural language processing, and that they are created through a multistage training process. In that process, very large and diverse collections of text are assembled and processed so that a neural network can identify broad statistical associations across the collection that are predictive of language usage patterns. Those associations include syntax, semantics, and other properties of language that emerge only after the model has processed a very large and diverse collection of data.

433. A pre-trained LLM therefore serves a wholly different purpose from that of any individual text used in training. Rather than being read for the value of information, expression of opinions, or entertainment of a particular source, the LLM functions as a general technology that can be applied across a wide variety of tasks. The model's utility arises from statistical associations that are learned across the complete distribution of the training data, and not from any individual text or isolated subset of texts.

434. Additional stages of development, including supplementary pre-training, fine-tuning, and post-training alignment, further develop and refine the model's capabilities, specifically shaping its behavior so that it can follow instructions and exhibit desired characteristics such as helpfulness and safety. Across these

[redacted]

training stages, text is used functionally to develop and align the model without storing or preserving the original text of any individual source.

435. The outputs generated by properly trained LLMs are novel. The objective of training is to enable the model to generalize to new and previously unseen contexts rather than to memorize particular training examples. More specifically, the outputs are generated at inference time by sampling from probability distributions that are dynamically conditioned on the user's prompt, the prior generated tokens, and any of the additional context supplied to the model. Additionally, the textual training material is not stored by the model during training, nor is it ordinarily reproduced during inference.

436. When an LLM-based product uses retrieval-augmented generation, or grounding, the model may also incorporate external data during inference. This improves factual accuracy in the model's responses by allowing the system to rely on current, context-specific information, in addition to the broad statistical associations learned during training, which includes information that may post-date the model's training itself.

437. My brief discussion of Copilot provides the technical background and context necessary for the empirical analysis that follows. A more detailed account of the development and operation of the product is, as I understand from Counsel, addressed by Microsoft's fact witnesses. For purposes of this declaration, the relevant point is that the product background provides the context needed to evaluate the output data generated by consumer use of Copilot.

438. Lastly, the sample I define from the Chat Log Data provides a sufficient basis to evaluate the general, real-world usage patterns of consumer Copilot. Considering how those data are stored and structured, the fields contained in the dataset, and the statistical analyses described above, the appropriate approach is to draw a simple random sample at the conversation level using conversation ID for selection. On that basis, an appropriate minimum sample size for analyzing general usage patterns is 30,000 conversations.

[redacted]

439. Plaintiffs' Experts' reports rest on a set of technical conflations. They treat next-token prediction as an instruction to reproduce training text, model weights as if they were compressed document repositories, grounding as if it were user-facing copying, and filtered n-gram or domain matches as if they establish population-level regurgitation. These characterizations are not consistent with the way LLMs are trained, represented, or used during inference.

440. As I explain above, a useful LLM is developed to generalize. Its weights reflect distributed statistical relationships learned across a broad training distribution. The model may learn facts, language patterns, genres, and associations from the data on which it is trained, and memorization can occur in very limited circumstances, but those points do not imply that the model stores Plaintiffs' works as retrievable copies or that reproduction is the ordinary function of the model.

441. Plaintiffs' Experts also overstate the technical importance of specific content categories. News and books may contain useful training signals, but the cited materials do not show that Plaintiffs' works, or copyrighted news and books as categories, are uniquely necessary to LLM capability. This is also true of Deucalion, where the small number of identified Plaintiff-domain references, without an appropriate denominator and causal analysis, does not show that Plaintiff content drove performance improvements that are otherwise consistent with post-training on grounding examples.

442. The Copilot output analyses confirm the same point empirically. Plaintiff's Expert's own experiments and analyses identify very low rates of alleged regurgitation, and those rates are inflated by methodological choices such as the use of a keyword/domain-filtered subset, treatment of [redacted] as user-facing output, short or inconsistent n-gram thresholds, many-to-one matching across works, and unsupported extrapolation. Domain hits, snippets, impressions, and redaction counts similarly do not establish that Copilot output copied substantial portions of Plaintiffs' works. Similarly, the presence of redactions does not hinder the ability for experts to run n-gram analyses over the Copilot output logs.

[redacted]

443. I disagree with Plaintiffs' Expert's characterization of LLM training, grounding, and Copilot output behavior. The materials and analyses described in this declaration are consistent with broad distributional learning, dynamic generation, limited and atypical memorization, and rare output regurgitation, and not with a system designed or operating to store and reproduce Plaintiffs' works.

444. Dr. Goldstein's analyses rest on flawed questions and misleading presentations that inflate the apparent prevalence of news and "pink slime" content retrieved and regurgitated by Copilot. Further, Dr. Goldstein's definition of a "leak" is a misnomer that conflates regurgitation with the normal operation of grounding, which retrieves information from the web via search.[358]

445. Plaintiffs' Experts rely on academic literature to support the claims that generalization requires memorization in deep learning systems and that training a language model is equivalent to compressing the training data.[359] The literature that is used to support these claims rely on theoretical models and stylized assumptions which are not valid for modern LLMs, thus invalidating their claims.

446. In his supplemental analysis, Dr. Goldstein adopts a more conservative analysis approach in comparison to Dr. Goldstein's previous analyses.[360] This more conservative approach utilized in Dr. Goldstein's supplemental analysis yields results similar to those put forward by other Plaintiffs' Experts and to my own analysis.

447. In his supplemental analysis, Dr. Goldstein claims that the filters applied to create the 45,000 conversation sample for Dr. Bail's categorization exercise are not appropriate.[361] However, the filters selected for the 45,000 conversation

[358] Hurst Decl. Ex. 234, Goldstein Rebuttal Report, ¶¶77-82, 109-120.
[359] Hurst Decl. Ex. 248, Wenger Rebuttal Report, pp.7-8; Hurst Decl. Ex. 252, Zhao Rebuttal Report, ¶¶23-25.
[360] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶27-29; Hurst Decl. Ex. 231, Goldstein Opening Report, ¶398.
[361] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶20.

[redacted]

sample are well suited for the categorization of how the general U.S. public uses Copilot.

448. Lastly, in his supplemental analysis, Dr. Goldstein provides examples of News Plaintiff content in Copilot logs.[362] My opinion stated above holds: snippets of content in the grounding field is part of the normal operation of the retrieval system. Such instances are not to be confused with regurgitation.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct. Executed on this 3rd day of September 2026, at New Haven, Connecticut.



John D. Lafferty, Ph.D.

[362] Hurst Decl. Ex. 235, Goldstein Supplemental Report, ¶¶53-58.

[redacted]

APPENDIX A

John Lafferty

- Address:
- Department of Statistics and Data Science
- Yale University
- 219 Prospect Street
- New Haven, CT 06520

Phone: 203.432.5216

Email: john.lafferty@yale.edu

A. Research Interests

- Machine learning; language modeling; foundation models, nonparametric statistics and high dimensional data analysis; computational modeling for neuroscience and cognition; graphical models; statistical learning theory, semisupervised learning; structured prediction, convex and nonconvex optimization, approximate inference methods.

B. Academic and Industrial Positions

- 2021–Present, Yale University
  - Associate Director, Wu Tsai Institute at Yale University
  - Director, Center for Neurocomputation and Machine Intelligence
- 2017–Present, Yale University
  - John C. Malone Professor of Statistics and Data Science
  - Secondary appointment, Department of Computer Science
- 2013–2019, Toyota Technological Institute at Chicago
  - Adjoint Professor
- 2011–2017, University of Chicago

[redacted]

- Louis Block Professor of Statistics and Computer Science

- 2011–2013, Carnegie Mellon University
  - Adjunct Professor, Computer Science and Machine Learning
- 2010–2011, Voleon Capital Management, Berkeley, CA
  - Principal Research Scientist
- 2004–2011, Carnegie Mellon University
  - Professor of Computer Science, Machine Learning, and Statistics
  - Co-Director, Ph.D. Program in Machine Learning (2003–2008)
- 1998–2004, Carnegie Mellon University
  - Associate Professor of Computer Science
  - Affiliated faculty, Program in Algorithms, Combinatorics and Optimization
- 1994–1997, Carnegie Mellon University
  - Assistant Research Professor, Computer Science Department
- 1988–1994, IBM Thomas J. Watson Research Center, Yorktown Heights, NY
  - Research Staff Member
- 1986–1987, Harvard University
  - Assistant Professor, Benjamin Pierce Lecturer on Mathematics

C. Visiting Positions

- 2016, January–March, Visiting Researcher, Microsoft Research, Cambridge MA
- 2007–2008, Visiting Professor, University of California, San Diego, Computer Science Department
- 2007, September–October, Visiting Researcher, Microsoft Research, Beijing, China
- 2001, January–June, Visiting Professor, University of California, Berkeley, Department of Statistics

[redacted]

- 1996, June-August, Visiting Professor, Dartmouth College, Department of Computer Science

D. Education

- 1982–1986, Princeton University
- Ph.D. in Mathematics, 1986, M.A. in Mathematics, 1984
- Member of the Program in Applied and Computational Mathematics
- Thesis advisor: Edward Nelson
- 1981–1982, Massachusetts Institute of Technology
- Special Student, graduate and undergraduate work in mathematics and statistics
- Beinecke Scholar
- 1978–1981, Middlebury College
- B.A. 1982 summa cum laude with highest honors in Mathematics

E. Honors

- Bahadur Lecturer, University of Chicago, May 2025
- https://stat.uchicago.edu/events/bahadur-lectures
- Lex Dixon ’63 Prize for Teaching Excellence, Yale University, 2024.
- Challis Award Lecture, University of Florida (stat.ufl.edu/challis-lectures), 2018.
- Pillai Memorial Lecture, Purdue University, 2018.
- IEEE Micro Honorable Mention, for paper “Combining Machine Learning and Control to Manage Computing System Complexity and Dynamics”
- Title: Combining Machine Learning and Control to Manage Computing System Complexity and Dynamics
- 2016 Test of Time Award (best 10 year paper) from the International Conference on Machine Learning, for paper “Dynamic topic models,” with David Blei

[redacted]

- 2015, IMS Medallion Lecture, Joint Statistical Meetings, Seattle
- 2014 Test of Time Award (best 10 year paper) from SIGIR for paper "Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval," with Cheng Zhai and William Cohen
- 2013 Test of Time Award (best 10 year paper) from the International Conference on Machine Learning, for paper "Semi-supervised learning using Gaussian fields and harmonic functions," with Xiaojin Zhu and Zoubin Ghahramani
- 2012 Test of Time Award (best 10 year paper) from the International Conference on Machine Learning, for paper "Diffusion kernels on graphs and other discrete input spaces," with Risi Kondor
- 2011 Test of Time Award (best 10 year paper) from the International Conference on Machine Learning, for paper "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," with Andrew McCallum and Fernando Pereira
- 2006, IEEE Fellow, for contributions to statistical pattern recognition and statistical language processing
- 1998, 1999, IBM University Partnership Award
- 1991, IBM Research Division Award
- 1985, Nominated for Princeton University Graduate Teaching Award
- 1982, Princeton University Merit Award
- 1981-1983, Beinecke Scholarship, full tuition and stipend support during senior year at MIT, and first two years of graduate school at Princeton

F. Professional Service

- 2020–2024, co-Editor-in-Chief, IMA/ACM Transactions on Data Science.
- 2015–2018, Invited member of NORC External Advisory Committee on Statistics, Machine Learning, and High Performance Computing, Chicago IL

[redacted]

- 2015–2022, Institute for Mathematical Statistics (IMS) Committee on Special Lectures
- 2016–2018, Institute for Mathematical Statistics (IMS) Committee on Publications (chair)
- 2012–2016, Elected member, Committee on Applied and Theoretical Statistics (CATS), National Research Council of the National Academy of Sciences
- 2013–present, Member, Advisory Board, NIPS Foundation
- 2010–2013, Member, Executive Board, NIPS Foundation
- 2010, General Co-Chair, with Chris Williams, Neural Information Processing Systems (NIPS) Conference
- 2009, Program Co-Chair, with Chris Williams, Neural Information Processing Systems (NIPS) Conference

G. Editorial Service

- 2017–2019, Associate Editor, Journal of the American Statistical Association
- 2015–2019, Associate Editor, Bernoulli
- 2011-2020, Editorial Board, Journal of Machine Learning Research
- 2006–2012, Associate Editor, Electronic Journal of Statistics
- 2000-2011, Action Editor, Journal of Machine Learning Research
- 2000-2001, Associate Editor, Machine Learning
- 1994–1997, Associate Editor, Computational Linguistics

H. University Service, Yale University

- 2025–, Member of Provost's AI Steering Committee
- 2025–, Co-Director of Peter Salovey and Marta Moret Data Science Fellows Program
- 2024–, Executive Board, Yale Institute for Policy Studies

[redacted]

- 2021–, Associate Director, Wu Tsai Institute

I. University Service, University of Chicago

- 2016–2017, Chief Academic Officer Search Committee, Toyota Technological Institute at Chicago
- 2014–2016, Member of Undergraduate College Council
- 2014–2015, Chair of Standing Committee on Academic Fraud
- 2013–2014, Member of Standing Committee on Academic Fraud
- 2013–2014, Member of President's Science Group Committee on Computation Strategy
- 2012–2013, Member of Provost's Committee on On-Line Education
- 2012, Member of University Core Group on Data Science

J. University Service, Carnegie Mellon University

- 2004–2006, Member of Faculty Review Committee
- 2003–2008, Co-Director (with Steve Fienberg), Ph.D. Program in Machine Learning
- 2000–2001, Member of University Faculty Senate
- 2000, Chair of Graduate Admissions Committee, Computer Science Department
- 1997–2000, Organizer of Immigration Course Research Symposium, School of Computer Science

K. Ph.D. Students Graduated

- Chris Xu, 2023 (software engineer, Google)
- Ganlin Song, 2021 (software engineer, Facebook)
- Zifan Li, 2021 (software engineer, Facebook)
- Natalie Doss, 2020 (research scientist, Two Sigma)

[redacted]

- Dana Yang, 2019 (assistant professor, Cornell University)
- Qinqing Zheng, 2018 (Facebook, Wharton School post-doc)
- Nikita Mishra, 2017 (Apple)
- Yuancheng Zhu, 2016 (Wharton School, Renaissance Technologies)
- Zhe Liu, 2016 (Google, Facebook)
- Min Xu, 2015 (Wharton School, Rutgers University)
- Dinah Shender, 2014 (Google)
- Eric Janofsky, 2014 (Conversant, Uber)
- Han Liu, 2011 (Princeton University, Northwestern University)
- Mugizi Rwebangira, 2008 (Howard University)
- Doug Vail, 2008 (Google)
- Pradeep Ravikumar, 2007 (University of Texas, Carnegie Mellon University)
- Leonid Kontorovich, 2007 (Ben Gurion University)
- Jerry Zhu, 2006 (University of Wisconsin, Madison)
- Paul Bennett, 2006 (Microsoft Research)
- Guy Lebanon, 2005 (LinkedIn)
- ChengXiang Zhai, 2003 (University of Illinois, Urbana-Champaign)
- Peter Venable, 2003 (Google)
- Adam Berger, 2001 (UPMC Enterprises)

L. Postdoctoral Fellows Advised

- Baohua Zhou, 2019–2024 (joint with Damon Clark)
- Kriste Krstovski, 2016–2019 (Columbia University and University of Chicago)
- Sabyasachi Chatterjee, 2014–2016 (University of Chicago, now University of Illinois)

[redacted]

- Minhua Chen, 2012–2013 (Amazon)
- Shuheng Zhou, 2008–2009 (University of Michigan)
- David Blei, 2007–2008 (Princeton University, now Columbia University)
- Ramesh Nallapati, 2006 (Stanford University, now SRI International)

M. Special Events Organized

- Co-Organizer, with Jon Cohen, "WTI Workshop on Abstraction and Relational Reasoning" Wu-Tsai Institute, November 16-17, 2023, https://wti.yale.edu/event/2023-11/wti-workshop-abstraction-and-relational-reasoning.
- Co-Organizer, with Stephane Bonhomme, Lars Peter Hansen and Thibaut Lamadon, "Machine Learning: What's in it for Economics?" Becker-Friedman Institute for Research in Economics, University of Chicago, September 23-24, 2016, https://bfi.uchicago.edu/events/machine-learning-what%E2%80%99s-it-economics.
- Co-Organizer, with Jian Ding, Peter McCullagh, and Steve Stigler, "A Celebration of Statistics at Chicago," workshop on the occasion of the 60th anniversary of the Department, May 13-14, 2016, http://www.stat.uchicago.edu/frs60/index.shtml.
- Summer REU project "Learning Polynomials, Graphs and Densities," the University of Chicago Theory Center, http://theorycenter.cs.uchicago.edu/REU/2014/projects.php, 2014.
- Co-chair, with Raghu Ramakrishnan, Microsoft Research, "Training Students to Extract Value from Big Data," under the Committee on Applied and Theoretical Statistics (CATS) of the National Research Council, April 11-12, 2014. Report: http://bit.ly/1DuCDAu
- Co-Organizer, "SAMSI Program on Low-dimensional Structure in High-dimensional Systems," 2013-2014.

[redacted]

- Co-Organizer, with Xiaotong Shen and Joseph Verducci, AMS-IMS-SIAM 2007 Joint Summer Research Conference on "Machine Learning and Statistics: Prediction and Discovery," Snowbird, Utah, June 2007.
- Co-Organizer, with Xiaotong Shen and Joseph Verducci, AMS-IMS-SIAM 2006 Joint Summer Research Conference on "Machine Learning and Statistics: Prediction and Discovery," Snowbird, Utah, June 2006.
- Co-Organizer, with Colin Cooper, Christos Faloutsos, and Alan Frieze, NSF-Aladdin Workshop on "Web Structure and Algorithms," Carnegie Mellon University, April, 2004.
- Co-Organizer, with Avrim Blum, Jon Kleinberg, Jianbo Shi, Eva Tardos, and Ramin Zabih, NSF-Aladdin Workshop on "Graph Partitioning in Vision and Learning," Carnegie Mellon University, January 2003.
- Co-organizer, with Bruce Croft and Jamie Callan, Workshop on "Language Modeling and Information Retrieval," Pittsburgh, PA, May 31–June 1, 2001.

N. Invited Talks

- "Models of Abstraction and Reasoning," Columbia University, November 2025.
- "CoT Information," Renaissance Technologies, Long Island, NY, June 2025.
- "CoT Information," Plenary keynote talk, New England Statistics Symposium, June 2025.
- "Machine learning models of abstraction," Flatiron Institute, New York, NY, April 2025.
- "Machine learning models of abstraction," Two Sigma, New York, NY, April 2025.
- "Compressed Sensing and Repair of Overparameterized Models," inter-departmental seminar, Departments of Psychology and Statistics and Data Science, Yale, November 2020.
- "The Unreasonable Effectiveness of Gradient Descent," Baidu AI Seminar, July 2020.

[redacted]

- "The Unreasonable Effectiveness of Gradient Descent," Invited plenary keynote, Conference on Algorithmic Learning Theory, February 2020.
- "Structure and Adaptivity in Optimization, Learning and Inference," 50th Journées de Statistique, Paris France, June 2018.
- "'Fruit Flies' in Statistics and Machine Learning," University of Florida, Challis Lectures April 2018.
- "'Fruit Flies' in Statistics and Machine Learning," Pillai Lecture, Purdue University. April 2018.
- "Structure and Adaptivity in Optimization, Learning and Inference," University of Florida, Challis Lectures April 2018.
- "Superadaptive Optimization (and Inference)," C. V. Starr Lectureship Series, Brown University, April 2017.
- "Superadaptive Optimization (and Inference)," Joint Statistics/Computer Science Colloquium, Harvard University, March 2017.
- "Constrained and Localized Estimation and Optimization," Distinguished Lecture, Ben-Gurion University, February 2017.
- "Constrained and Localized Nonparametric Estimation and Optimization," Cambridge Statistical Laboratory, November 2016.
- "The Complexity of Minimizing Individual Convex Functions," TTIC Seminar, June 2016.
- "Statistical Learning under Communication and Shape Constraints," Northwestern University, EECS Colloquium, June 2016.
- "Statistical Learning under Communication and Shape Constraints," University of Southern California, Computer Science Colloquium, May, 2016.
- "Statistical Learning under Communication Constraints," Microsoft-MIT Machine Learning Seminar, March, 2016.

[redacted]

- "On Shape-Constrained Estimation," MIT Center for Statistics Seminar, February, 2016.
- "Nonparametric Estimation in High Dimensions," Illinois Statistics Symposium: A Celebration of the Department's 30th Anniversary, November 2015.
- "Quantized Nonparametric Estimation," Carnegie Mellon University, Machine Learning Seminar, June 2015.
- "Statistical Learning in High Dimensions," Distinguished Lecture, Boston University Data Science Initiative, November 2015.
- "Scalable First-Order Algorithms for Rank Minimization," 53rd Annual Allerton Conference on Communication, Control, and Computing October, 2015.
- "Storage and Shape Constraints in Nonparametric Estimation," Inaugural Seminar Series, Princeton Center for Statistics and Machine Learning, May 2015.
- "Information Theoretic Perspectives in Statistical Estimation," University of Illinois, Urbana-Champaign, CSL Student Conference, February 2015.
- "High Dimensional Convex Function Estimation," Optimization and Statistical Learning, Les Houches, January 2015.
- "Variable Selection in High Dimensional Convex Regression," Duke-UCL Workshop on Sensing and Analysis of High-Dimensional Data, September 2014.
- "Variable Selection in High Dimensional Convex Regression," Invited Session on Shape-Restricted Estimation, Joint Statistical Meetings, Boston, August 2014.
- "The Interplay Between Computation and Estimation," Prospects in Applied Mathematics, University of Chicago, http://www.math.uchicago.edu/~bowei/pam2014.pdf, October 2014.
- "Regression as Compression (and Vice Versa)," Plenary Lecture, Princeton Conference on Information Sciences and Systems, March 2014.
- "Variants on Nonparametric Additive Models for Regression and Graphical Modeling," Department of Statistics, Northwestern University, May 2013.

[redacted]

- "Exchanging Bips for Flops and Bits: Computational Tradeoffs in Statistical Learning," Duke University Machine Learning Seminar Series, November 2013.
- "Exchanging Bips for Flops and Bits: Computational Tradeoffs in Statistical Learning," Yale University Statistics Seminar, November 2013.
- "Exchanging Bips for Flops and Bits: Computational Tradeoffs in Statistical Learning," The School of Informatics and Computing Computer Science Colloquium Series, Indiana University November 2013.
- "Exchanging Bips for Flops and Bits: Computational Tradeoffs in Statistical Learning," Department of Statistics Seminar, University of Wisconsin, Madison, October, 2013.
- "Localized Minimax Complexity of Stochastic Convex Optimization," 51st Annual Allerton Conference on Communication, Control, and Computing, October, 2013
- "Graph Estimation," University of Illinois, Urbana-Champaign, Machine Learning Seminar, November 2012.
- "Variations on Nonparametric Additive Models: Computational and Statistical Aspects," 3rd Princeton Day of Statistics, September, 2012.
- "Variations on Nonparametric Additive Models: Computational and Statistical Aspects," Purdue University, Department of Statistics and Department of Computer Science Colloquium, August, 2012.
- "Matrix Sparse Coding," Duke Workshop on Sensing and Analysis of High-Dimensional Data, July 2011.
- "Learning Functions and Graphs in High Dimensions," Invited Plenary Talk, Conference on AI and Statistics (AISTATS), 2010, Sardinia, Italy. May 2010.
- "Nonparametric Graphical Models," Machine Learning Seminar Series, Georgia Institute of Technology, May 2010.
- "Nonparametric Graphical Models," Colloquium, Google New York Research Lab, April 2010.

[redacted]

- "Three Rivers in Machine Learning: Data, Risk and Computation," LIDS Colloquium, Laboratory for Information and Decision Systems, MIT, January 2010.
- "Three Rivers in Machine Learning: Data, Risk and Computation," Distinguished Lecture Series, Department of Computer Science, Columbia University, January 2010.
- "Estimating High Dimensional Graphs," Colloquium Series, Renaissance Technologies, May 2009.
- "Functional Sparsity," Seminar, Department of Statistics, University of Chicago, January 2008.
- "Functional Sparsity," Applied Mathematics Colloquium, Princeton University, November 2007.
- "Challenges in Statistical Machine Learning," Distinguished Speaker Series in Artificial Intelligence and Machine Learning, University of California, Irvine, Center for Genomics and Bioinformatics, March 2007.
- "Challenges in Statistical Machine Learning," Distinguished Lecture Series, Toyota Technology Institute, March, 2007.
- "Exploiting Sparsity and Structure in Statistical Learning," Center for Imaging Science, Johns Hopkins University, February, 2007.
- "Structured Prediction," World Congress of the Bernoulli Society for Mathematical Statistics and Probability, and Annual Meeting of the Institute of Mathematical Statistics, Barcelona, July 2004.
- "Data and Structure in Machine Learning," Distinguished Lecture Series, Computer and Information Sciences Department, University of Pennsylvania, November, 2003.
- "Random Fields, Graph Kernels, and Semi-Supervised Learning," MIT AI Colloquium, November, 2003.
- "Information Theoretic Methods in Approximate and Semi-Supervised Bayesian Inference," Invited Plenary Speaker, Canadian Workshop on Information Theory, Waterloo, Ontario, May, 2003.

[redacted]

O. Invited Tutorials and Short Course

- Short course, Columbia University, Department of Statistics, "Statistical Learning in High Dimensions," summer semester, 2016.
- Short course, "Machine Learning and Large-Scale Data Analysis," NORC at the University of Chicago, summer 2012.
- Short course, "Statistical Machine Learning for High Dimensional Data," Vietnamese Institute for Advanced Study in Mathematics (VIASM), Hanoi, Vietnam, summer 2012.
- Short course, with Larry Wasserman (Carnegie Mellon University), "Statistical Machine Learning for High Dimensional Data," Goldman Sachs Lecture Series, 2011.
- Tutorial, "Challenges in Statistical Machine Learning," Workshop on Information Theory and Applications, University of California, San Diego, February 2007.
- Tutorial, "Exponential Models and Maximum Entropy for Probabilistic Modeling," Conference on Uncertainty in Artificial Intelligence (UAI), Seattle, WA, August 2001.
- Tutorial, "Probabilistic Language Models," Annual conference on Neural Information Processing Systems (NIPS), 1999.
- Tutorial, "Statistical Methods in Natural Language Processing," with Lillian Lee (Cornell University), Sixteenth National Conference on Artificial Intelligence (AAAI), Orlando, FL, 1999.
- Tutorial, "Statistical Methods in Natural Language Processing," with Lillian Lee (Cornell University), Fifteenth National Conference on Artificial Intelligence (AAAI), Madison, WI, 1998.

P. Publications

- "Confidence Intervals for Linear Models with Arbitrary Noise Contamination," Chao Gao, John Lafferty, and Dong Xie, Biometrika, to appear, 2026.

[redacted]

- “Information-Computation Tradeoffs for Noiseless Linear Regression with Oblivious Contamination,” Ilias Diakonikolas, Chao Gao, Daniel Kane, John Lafferty, Ankit Pensia, arXiv:2510.10665; NeurIPS 2025.
- “CoT Information: Improved Sample Complexity under Chain-of-Thought Supervision,” Awni Altabaa, Omar Montasser, and John Lafferty, arXiv:2505.15927; NeurIPS 2025.
- “Disentangling and integrating relational and sensory information in transformer architectures,” Awni Altabaa and John Lafferty, ICML 2025.
- “Approximation of relation functions and attention mechanisms,” Awni Altabaa and John Lafferty, to appear in Barron Festschrift, 2025.
- “Images with harder-to-reconstruct visual representations leave stronger memory traces,” Qi Lin, Zifan Li, John Lafferty and Ilker Yildirim, Nature Human Behaviour, volume 8, pages 1309–1320, 2024.
- “Learning hierarchical relational representations through relational convolutions,” Awni Altabaa and John Lafferty, ICLR, 2024.
- “The Relational Bottleneck as an Inductive Bias for Efficient Abstraction” Taylor W. Webb, Steven M. Frankland, Awni Altabaa, Kamesh Krishnamurthy, Declan Campbell, Jacob Russin, Randall O’Reilly, John Lafferty, Jonathan D. Cohen, Trends in Cognitive Science, Volume 28, Issue 9, September 2024, Pages 829-843.
- “Abstractors and relational cross-attention: An inductive bias for explicit relational reasoning in Transformers," Awni Altabaa, Taylor Webb, Jonathan Cohen, John Lafferty, International Conference on Learning Representations, ICLR 2024.
- “Shallow neural networks trained to detect collisions recover features of visual loom-selective neurons," Baohua Zhou, Zifan Li, Sunnie Kim, John Lafferty, Damon A. Clark, eLife, 2022. https://elifesciences.org/articles/72067.
- “Convergence and alignment of gradient descent with random backpropagation weights,” Ganlin Song, Ruitu Xu, and John Lafferty, Proceedings of Neural Information Processing Systems, 2021.

[redacted]

- "Model repair: Robust recovery of over-parameterized statistical models," Chao Gao and John Lafferty, arXiv:2005.09912; submitted.
- “Surfing: Iterative optimization over incrementally trained eep networks," Ganlin Song, Zhou Fan, and John Lafferty, Proceedings of Neural Information Processing Systems, 2019.
- “Prediction rule reshaping,” Matt Bonakdarpour, Sabyasachi Chatterjee, Rina Foygel Barber, and John Lafferty, International Conference on Machine Learning (ICML), 2018.
- “Distributed nonparametric regression under communication constraints,” Yuancheng Zhu and John Lafferty,
- Sabyasachi Chatterjee and John Lafferty, “Adaptive risk bounds in unimodal regression,” arXiv:1512.02956, Bernoulli, Volume 25, Number 1 (2019), 1–25.
- Nikita Mishra, Connor Imes, Henry Homann, and John Lafferty, “Controlling AI Engines in Dynamic Environments,” SysML 2018.
- Sabyasachi Chatterjee and John Lafferty, “Denoising flows on trees,” IEEE Trans. Info. Theory, Vol. 64, No. 3, 2018.
- Chao Gao and John Lafferty, “Testing for global network structure using small subgraph statistics,” arXiv:1710.00862, 2017; submitted for publication.
- Chao Gao and John Lafferty, “Testing network structure using relations between small subgraph probabilities,” arXiv:1704.06742, 2017.
- Nikita Mishra, John Lafferty, and Henry Hoffmann, “ESP: A Machine Learning Approach to Predicting Application Interference,” Proceedings of International Conference on Autonomic Computing, 2017.
- Fan Yang, Rina Foygel Barber, Prateek Jain and John Lafferty, “Selective inference for group-sparse linear models,” NIPS 2017; arXiv:1607:08211.
- Yuancheng Zhu, Sabyasachi Chatterjee, John Duchi, John Lafferty, “Local minimax complexity of stochastic convex optimization,” NIPS 2017; arXiv:1605.07596.

[redacted]

- Qinqing Zheng and John Lafferty, "Convergence analysis for rectangular matrix completion using Burer-Monteiro factorization and gradient descent," arXiv:1605.07051.
- Yuancheng Zhu and John Lafferty, "Quantized estimation over Sobolev ellipsoids," Information and Inference, 2017.
- Qinqing Zheng and John Lafferty, "A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements," Advances in Neural Information Processing Systems 28, 2015, arXiv:1506.06081.
- Min Xu, Minhua Chen and John Lafferty, "Faithful variable selection for high dimensional convex regression," arXiv:1411.1805, to appear in Annals of Statistics.
- Nikita Mishra, Harper Zhang, John Lafferty and Henry Hoffman, "A graphical model approach to minimizing energy under performance constraints," International Conference on Architecural Support for Programming Languages and Operating Systems (ASPLOS), 2015.
- Yuancheng Zhu and John Lafferty, "Quantized estimation of Gaussian sequence models in Euclidean balls," Advances in Neural Information Processing Systems 28, 2014.
- Zhe Liu and John Lafferty, "Blossom tree graphical models," Advances in Neural Information Processing Systems 28, 2014.
- Dinah Shender and John Lafferty, "Computation-risk tradeoffs for covariance-thresholded regression," International Conference on Machine Learning, JMLR W&CP Vol. 28(3): 756–764, 2013.
- Alfredo Kalaitzis, John Lafferty, Neil Lawrence, and Shuheng Zhou, "The bigraphical lasso," International Conference on Machine Learning, JMLR W&CP 28(3): 1229–1237, 2013.
- Sivaraman Balakrishnan, Kriti Puniyani and John Lafferty, "Sparse additive functional and kernel CCA," Proceedings of the 29th International Conference on Machine Learning (ICML-12), New York, NY, pp. 911-918, 2012.

[redacted]

- Rina Foygel, Michael Horrell, Mathias Drton, and John Lafferty, "Nonparametric reduced rank regression," Advances in Neural Information Processing Systems, 25, pp 1637–1645, 2012.
- Han Liu, John Lafferty, Larry Wasserman, "Exponential concentration for mutual information estimation with application to forests," Advances in Neural Information Processing Systems 25, pp 2546–2554, 2012.
- Han Liu, Fang Han, Ming Yuan, John D. Lafferty and Larry A. Wasserman, "High-dimensional semiparametric Gaussian copula graphical models," Ann. Statist. Vol. 40, No. 4, 2293–2326, 2012.
- John Lafferty, Han Liu, and Larry Wasserman, "Sparse nonparametric graphical models," Statistical Science, Vol. 27, No. 4, 519–537, 2012.
- Han Liu, John Lafferty, and Larry Wasserman, "The nonparanormal: Semiparametric estimation of high dimensional undirected graphs," Journal of Machine Learning Research, Vol. 10, pp. 2295–2328, 2009.
- Han Liu, Min Xu, Haijie Gu, Anupam Gupta, John Lafferty and Larry Wasserman, "Forest density estimation," Journal of Machine Learning Research, Vol. 12, pp. 907–951, 2011.
- Mladen Kolar, John Lafferty and Larry Wasserman, "Union support recovery in multi-task learning," Journal of Machine Learning Research, Vol. 12, pp. 2415–2435, 2011.
- Pradeep Ravikumar, Martin Wainwright and John Lafferty, "High dimensional Ising model selection using l_1 regularized logistic regression," Ann. Statist. Volume 38, Number 3 (2010), 1287–1319.
- Shuheng Zhou, John Lafferty and Larry Wasserman, "Time varying undirected graphs," Machine Learning, Volume 80, Numbers 2-3, September 2010, pp. 295-319.
- David Blei and John Lafferty, "Visualizing topics with multi-word expressions," http://arxiv.org/abs/0907.1013, 2009.

[redacted]

- Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman, "Sparse additive models," Journal of the Royal Statistical Society: Series B, Volume 71, part 5, pp. 1009–1030, 2009.
- Shuheng Zhou, John Lafferty and Larry Wasserman, "Compressed and privacy sensitive sparse regression," IEEE Trans. Information Theory, Vol. 55, Number 2, pp. 846–866, February 2009.
- John Lafferty and Larry Wasserman, "Rodeo: Sparse, greedy nonparametric regression," Annals of Statistics, Vol. 36, No. 1, pp. 28–63, 2008.
- John Lafferty and Larry Wasserman, "Statistical analysis of semi-supervised regression," In Advances in Neural Information Processing Systems (NIPS), 2008.
- David Blei and John Lafferty. "Topic models," in A. Srivastava and M. Sahami, editors, Text Mining: Theory and Applications. Taylor and Francis, 2009.
- Yoshua Bengio, Dale Schuurmans, John lafferty, Chris K. I. Williams, and Aaron Culotta (Editors), Advances in Neural Information Processing Systems 22, 2009.
- John Lafferty and Larry Wasserman, "Comments on: Nonparametric inference with generalized likelihood ratio tests, nonparametric sparsity," Test, Volume 16, pp. 453–455, 2007.
- Han Liu, John Lafferty, and Larry Wasserman, "Sparse nonparametric density estimation in high dimensions using the Rodeo," Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS), 2007.
- David Blei and John Lafferty, "A correlated topic model of Science," Annals of Applied Statistics, Vol. 1, No. 1, 17–35, 2007.
- Ramesh Nallapati, William Cohen, Susan Ditmore, John Lafferty, and Kai Ung, "Multi-scale topic tomography," In proceedings of ACM Conference on Knowledge Discovery and Datamining (KDD), 2007.
- Ramesh Nallapati, William Cohen, and John Lafferty, "Parallelized variational EM for latent Dirichlet allocation: An experimental evaluation of speed and scalability," ICDM workshop on high performance data mining, 2007.

[redacted]

- Doug Vail, John Lafferty, and Manuela Veloso, "Feature selection in conditional random fields for activity recognition," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2007.
- Matus Telgarsky and John Lafferty, "Signal decomposition using multiscale admixture models," International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2007 April 2007, Vol 2, pp. 449-452.
- Joseph Verducci, Xiaotong Shen, John Lafferty, editors, Prediction and discovery: AMS-IMS-SIAM Joint Summer Research Conference on Machine and Statistical Learning, June 25-29, 2006, Snowbird, Utah. American Mathematical Society, Providence, R.I., 2007.
- Han Liu, John Lafferty, and Larry Wasserman, "Sparse nonparametric density estimation in high dimensions using the Rodeo," Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS), 2007.
- Doug Vail, Manuela Veloso, and John Lafferty, "Conditional random fields for activity recognition," International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2007.
- Noah Smith, John Lafferty, and Doug Vail, "Computationally efficient M-estimation of log-linear structure models," Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Prague, Czech Republic, 2007.
- David Blei and John Lafferty, "Dynamic topic models," Machine Learning: Proceedings of the Twenty-Third International Conference (ICML), pp. 113–120, 2006.
- Pradeep Ravikumar and John Lafferty, "Quadratic programming relaxations for metric labeling and Markov random field MAP estimation," Machine Learning: Proceedings of the Twenty-Third International Conference (ICML), 2006.
- John Lafferty and Larry Wasserman, "Challenges in statistical machine learning," Statistica Sinica, Vol. 16, 207–322, 2006.

[redacted]

- Xiaojin Zhu, Jaz Kandola, John Lafferty and Zoubin Ghahramani, "Graph kernels by spectral transforms," In Semi-Supervised Learning, eds. Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien, The MIT Press, 2006.
- ChengXiang Zhai and John Lafferty, "A risk minimization framework for information retrieval," Information Processing and Management, Volume 42, Issue 1, pp. 31–55, January 2006.
- David Blei and John Lafferty, "Correlated topic models," Advances in Neural Information Processing Systems (NIPS), 18, 2005.
- Xiaojin Zhu and John Lafferty, "Harmonic mixtures: Combining mixture models and graph-based methods for inductive and scalable semi-supervised learning," Machine Learning: Proceedings of the Twenty-Second International Conference, 2005.
- Maria-Florina Balcan, Avrim Blum, Patrick Choi, John Lafferty, Brian Pantano, Mugizi Rwebangira and Xiaojin Zhu, "Person identification in webcam images: An application of semi-supervised learning," ICML 2005 Workshop on Learning with Partially Classified Training Data, 2005.
- John Lafferty and Guy Lebanon, "Diffusion kernels on statistical manifolds," Journal of Machine Learning Research, Vol. 6, pp. 129–163, 2005.
- Pradeep Ravikumar and John Lafferty, "Preconditioner approximations for probabilistic graphical models," Advances in Neural Information Processing Systems (NIPS), 18, 2005.
- Xiaojin Zhu, Jaz Kandola, Zoubin Ghahramani, and John Lafferty, "Nonparametric transforms of graph kernels for semi-supervised learning," Advances in Neural Information Processing Systems (NIPS), 17, 2004.
- Pradeep Ravikumar and John Lafferty, "Variational Chernoff bounds for graphical models," Uncertainty in Artificial Intelligence, 2004.
- John Lafferty, Yan Liu and Xiaojin Zhu, "Kernel conditional random fields: Representation and clique selection," Machine Learning: Proceedings of the Twenty-first International Conference, 2004.

[redacted]

- Guy Lebanon and John Lafferty, "Hyperplane margin classifiers on the multinomial manifold," Machine Learning: Proceedings of the Twenty-first International Conference, 2004.
- Avrim Blum, John Lafferty, R. Reddy and Mugizi Rwebangira, "Semi-supervised learning using randomized mincuts," Machine Learning: Proceedings of the Twenty-first International Conference, 2004.
- ChengXiang Zhai and John Lafferty, "A study of smoothing methods for language models applied to information retrieval," ACM Transactions on Information Systems, Vol. 212, No. 2, pp. 197–214, 2004.
- Elena Erosheva, Stephen Fienberg and John Lafferty, "Mixed-membership models of scientific publications," Proc. Natl. Acad. Sci. USA. Vol. 101, Suppl. 1, pp 5220–5227, 2004.
- Alexander Gamburd, John Lafferty, and Daniel Rockmore, "Eigenvalue spacings for quantized cat maps," Journal of Physics A: Mathematical and General, Vol. 36, Special Issue: Random Matrix Theory, 2003.
- Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani, "Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions," in Proceedings of ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, 2003.
- ChengXiang Zhai, William Cohen and John Lafferty, "Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval," Proceedings of ACM SIGIR, pp. 10–17, 2003.
- Xiaojin Zhu, Zoubin Ghahramani and John Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," Machine Learning: Proceedings of the Twentieth International Conference (ICML), pp. 912-919, 2003.
- ChengXiang Zhai and John Lafferty, "A risk minimization framework for information retrieval," Proceedings of the ACM SIGIR 2003 Workshop on Mathematical/Formal Methods in IR, 2003.

[redacted]

- W. Bruce Croft and John Lafferty, editors, "Language Modeling for Information Retrieval," Kluwer International Series on Information Retrieval, Vol. 13, 2003.
- John Lafferty and ChengXiang Zhai, "Probabilistic relevance models based on document and query generation," In Language Modeling for Information Retrieval, W. Bruce Croft and J. Lafferty, editors, Kluwer International Series on Information Retrieval, Vol. 13, 2003.
- John Lafferty and Guy Lebanon, "Information diffusion kernels," Advances in Neural Information Processing Systems (NIPS), 15, 2002.
- Guy Lebanon and John Lafferty, "Conditional models on the ranking poset," Advances in Neural Information Processing Systems (NIPS), 15, 2002.
- Guy Lebanon and John Lafferty, "Boosting and maximum likelihood for exponential models," Advances in Neural Information Processing Systems (NIPS), 2002.
- Thomas Minka and John Lafferty, "Expectation-propagation for the generative aspect model," Uncertainty in Artificial Intelligence (UAI), 2002.
- ChengXiang Zhai and John Lafferty, "Two-stage language models for information retrieval," ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2002.
- Risi Kondor and John Lafferty, "Diffusion kernels on graphs and other discrete input spaces," Machine Learning: Proceedings of the Nineteenth International Conference (ICML), San Mateo, CA: Morgan Kaufmann, 2002.
- Guy Lebanon and John Lafferty, "Cranking: Combining rankings using conditional probability models on permutations," Machine Learning: Proceedings of the Nineteenth International Conference (ICML), San Mateo, CA: Morgan Kaufmann, 2002.
- ChengXiang Zhai and John Lafferty, "Model-based feedback in the language modeling approach to information retrieval," Proceedings of the 10th ACM International Conference on Information and Knowledge Management (CIKM), 2001.

[redacted]

- John Lafferty and Larry Wasserman, "Iterative Markov chain Monte Carlo computation of reference priors and minimax risk," Proceedings of Uncertainty in Artificial Intelligence (UAI), 2001.
- Stephen Della Pietra, Vincen Della Pietra, and John Lafferty, "Duality and auxiliary functions for Bregman distances," CMU Technical Report CMU-CS-01-109, 2001.
- John Lafferty and Chengxiang Zhai, "Document language models, query models, and risk minimization for information retrieval," 24th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01), 2001, pp. 111–119.
- ChengXiang Zhai and John Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," 24th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01), 2001, pp. 334–342.
- John Lafferty and ChengXiang Zhai, "Probabilistic IR models based on document and query generation," Proceedings of the Workshop on Language Modeling and Information Retrieval, Carnegie Mellon University, May 31–June 1, 2001.
- John Lafferty, Andrew McCallum and Fernando Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in Proceedings of the International Conference on Machine Learning, pp. 282–289, 2001.
- John Lafferty and Daniel Rockmore, "Codes and iterative decoding on algebraic expander graphs," Proceedings of International Symposium on Information Theory and Its Applications, ISITA 2000.
- Adam Berger and John Lafferty, "Information retrieval as statistical translation," In 1999 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), pp. 222–229, 1999.
- Kamal Nigam, John Lafferty and Andrew McCallum, "Using maximum entropy for text classification," in Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering, 1999.
- Adam Berger and John Lafferty, "The Weaver system for document retrieval," Proceedings of TREC-8, Gaithersburg, MD, 1999.

[redacted]

- John Lafferty, "Additive models, boosting and inference for generalized divergences," Proceedings of Twentieth Annual Conference on Computational Learning Theory (COLT), 1999.
- Doug Beeferman, Adam Berger and John Lafferty, "Statistical models for text segmentation," Machine Learning, special issue on Natural Language Learning, C. Cardie and R. Mooney, eds., 34(1-3), pp. 177-210, 1999.
- John Lafferty and Alexander Vardy, "Ordered binary decision diagrams and minimal trellises," IEEE Trans. Computers, Vol. 48, No. 9, pp. 971–986, Sept., 1999.
- Doug Beeferman, Adam Berger and John Lafferty, "CyberPunc: A lightweight punctuation annotation system for speech," In Inter. Conf. on Acoustics Speech and Signal Processing (ICASSP), Seattle, 1998.
- John Lafferty and Daniel Rockmore, "Level spacings for Cayley graphs," in Emerging Applications of Number Theory, D. Hejhal, F. Chung, J. Friedman, M. Gutzwiller, and A. Odlyzko, eds., IMA Volumes in Mathematics and its Applications, Vol. 109, 1998.
- Stephen Della Pietra, Vincent Della Pietra, and John Lafferty, "Inducing features of random fields," IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(4), April, pp. 380–393, 1997.
- Doug Beeferman, Adam Berger and John Lafferty, "Text segmentation using exponential models," In Proceedings of Empirical Methods in Natural Language Processing (EMNLP), Providence, RI, pp. 35–46, 1997.
- Doug Beeferman, Adam Berger and John Lafferty, "A model of lexical attraction and repulsion," In Proceedings of ACL-EACL Joint Conferences, Madrid, Spain, pp. 373–380, 1997.
- John Lafferty, Stephen Della Pietra and Vincent Della Pietra, "Statistical learning algorithms based on Bregman distances," In Proceedings of Canadian Workshop on Information Theory, Fields Institute, Toronto, Canada, pp. 77-80, 1997.

[redacted]

- Adam Berger and John Lafferty, "Probabilistic decoding of low-density Cayley codes," Canadian Workshop on Information Theory, Fields Institute, Toronto, Canada, pp. 11-14, 1997.
- John Lafferty and Daniel Rockmore, "Spectral techniques for expander codes," Proceedings of ACM Symposium on Theory of Computing (STOC), pp. 160–167, 1997.
- John Lafferty and Daniel Rockmore, "Spectral techniques for expander codes and generalized cyclic codes," in Proceedings of 1997 IEEE International Symposium on Information Theory, Ulm, Germany, 1997.
- Paul Placeway and John Lafferty, "Cheating with imperfect transcripts," In ICSLP'96: Fourth International Conference on Spoken Language Processing, 1996.
- Yeyi Wang, John Lafferty, and Alexander Waibel, "Word clustering using parallel spoken language corpora," In ICSLP'96: Fourth International Conference on Spoken Language Processing, 1996.
- John Lafferty and Bernhard Suhm. "Cluster expansions and iterative scaling for maximum entropy language models," In Maximum Entropy and Bayesian Methods, K. Hanson and R. Silver, eds., Kluwer Academic Publishers, 1996.
- Bernhard Suhm, John Lafferty, Martin Maier, Alexander Waibel, and Torsten Zeppenfeld, "A maximum entropy approach to topic dependent language modeling," In proceedings of the Johns Hopkins Workshop on Speech and Language, 1995.
- John Lafferty, "Gibbs-Markov models," Computing Science and Statistics, Vol. 27, pp. 370–377, 1996.
- Dennis Grinberg, John Lafferty and Daniel Sleator, "A robust parsing algorithm for link grammars," In Proceedings of the Fourth International Workshop on Parsing Technologies, 1995.
- John D. Lafferty and Charles S. Peskin, "Phase space density and fluid flow: Conservation laws and a Boltzmann equation associated with the stochastic Newton equation," In Stochastic Processes, Geometry, and Physics II, S. Albeverio, U. Cattaneo, and D. Merlini, eds., World Scientific, 1995.

[redacted]

- Stephen Della Pietra, Vincent Della Pietra, John Gillett, John Lafferty, Harry Printz, Lubos Ureš, "Inference and estimation of a long-range trigram model," in Proceedings of the Second International Colloquium on Grammatical Inference and Applications, Lecture Notes in Artificial Intelligence, Vol. 862, Springer-Verlag, pp. 78–92 1994.
- Adam Berger, Peter Brown, Stephen Della Pietra, Vincent Della Pietra, John Gillett, John Lafferty, Robert Mercer, Harry Printz, Lubos Ureš, "The Candide system for machine translation," Human Language Technology, Proceedings of the ARPA Workshop on Speech and Natural Language, Morgan Kaufman Publishers, 1994.
- Frederick Jelinek, John Lafferty, David Magerman, Robert Mercer, Adwait Ratnaparkhi, Salim Roukos, "Decision tree parsing using a hidden derivation model," Human Language Technology, Proceedings of the ARPA Workshop on Speech and Natural Language, Morgan Kaufman Publishers, 1994.
- John Lafferty, "A probabilistic model of link grammar," IBM Technical Disclosure Bulletin. Volume 37, No. 02B, pp. 99–105, 1994.
- John Lafferty and R. Mercer, "Automatic word classification using features of spellings," in proceedings of The 9th Annual Conference of the University of Waterloo Centre for the New OED and Text Research, Oxford University Press, Oxford, England, 1993.
- John Lafferty and Daniel Rockmore, "Numerical investigation of the spectrum for certain families of Cayley graphs," DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 10, pp. 63–73, 1993.
- John Lafferty, "Statistical training and analysis of grammars," in Statistically-Based Computer Analysis of English, E. Black and G. Leech, editors. Radopi Publishers, Amsterdam, 1993.
- John D. Lafferty, Review of Generalized LR Parsing, Masaru Tomita, editor. Prague Bulletin of Mathematical Linguistics, Vol. 59-60 pp. 129–133, 1993.
- Peter Brown, Frederick Jelinek, John Lafferty, Robert Mercer, and Salim Roukos, "Acquisition of language models from text," J. Acoust. Soc. Am., Vol. 92: 4-2, 1992.

[redacted]

- John Lafferty, Daniel Sleator, and Davy Temperley, "Grammatical trigrams: A probabilistic model of link grammar," Proceedings of the the AAAI Fall Symposium on Probabilistic Approaches to Natural Language, Cambridge, MA, October 1992.
- John Lafferty and Daniel Rockmore, "Fast Fourier analysis for SL_2 over a finite field and related numerical experiments," Experimental Mathematics, Vol. 1: pp. 115–139, 1992.
- Peter Brown, Vincent Della Pietra, Stephen Della Pietra, John Lafferty, Robert Mercer, "Analysis, statistical transfer, and synthesis in machine translation," Proceedings of the Fourth International Conference on Theoretical and Methodological Aspects of Machine Translation, pp. 83-100, 1992.
- John Lafferty, Yanlin Yu, and Weiping Zhang, "A direct geometric proof of the Lefschetz fixed point formulas," Trans. Amer. Math. Soc., 329(2) pp. 571–582, 1992.
- Frederick Jelinek, John Lafferty, and Robert Mercer, "Basic methods of probabilistic context-free grammars," in Speech Recognition and Understanding: Recent Advances, Trends, and Applications, P. Laface and R. De Mori, editors. Springer Verlag, Series F: Computer and Systems Sciences, Volume 75, 1992.
- Ezra Black, John Lafferty, and Salim Roukos, "Development, evaluation, and results for a broad-coverage probabilistic grammar of English-language computer manuals," in Proceedings of the ACL, 1992.
- Ezra Black, Frederick Jelinek, John Lafferty, Robert Mercer, and Salim Roukos, "Decision tree models applied to the labelling of text with parts-of-speech," in Proceedings of the DARPA Speech and Natural Language Workshop, Arden House, February 1992.
- Ezra Black, Frederick Jelinek, John Lafferty, David Magerman, Robert Mercer, and Salim Roukos, "Towards History-based Grammars: Using richer models for probabilistic parsing," in Proceedings of the DARPA Speech and Natural Language Workshop, Arden House, February 1992.

[redacted]

- Frederick Jelinek and John Lafferty, "Computation of the probability of initial substring generation by stochastic context-free grammars," Computational Linguistics, 17(3):315-323, 1991.
- Peter Brown, John Cocke, Stephen Della Pietra, Vincent Della Pietra, Frederick Jelinek, John Lafferty, Robert Mercer, and Paul Roossin, "A statistical approach to machine translation," Computational Linguistics, 16(2):79–85, 1990.
- John Lafferty, "The density manifold and configuration space quantization," Trans. Amer. Math. Soc., 305(2) pp. 699–741, 1988.

Q. Research Grants

- National Science Foundation, "Models and Mechanisms of Abstraction," PI, CISE Division, sub-award to Columbia AI Institute award, $100,000, 2023.
- National Science Foundation, "Generative Models for Complex Data: Inference, Sensing, and Repair," PI, Division of Mathematical Sciences, $250,000, 2020.
- JP Morgan AI Faculty Research Award, "Generation X: Learning and Inversion of Generative Networks for Tabular Data," PI, $150,000, 2019.
- National Science Foundation, "TRIPODS+X:RES: Investigations at the Interface of Data Science and Neuroscience," co-PI, $599,992, 2018.
- National Science Foundation, "RTG: Computational and Applied Mathematics in Statistical Science," $1,749,360, 2016.
- Alfred P. Sloan Foundation, "Extracting Mathematical Knowledge from the Scientific Literature: Statistical Machine Learning Models and Tools," $900,000, 2015.
- DARPA, "Proteus: Controlling Resource-Adaptive Embedded Software" (co-PI), $1,645,000, 2015.
- Facebook (AI/ML), unrestricted gift, $25,000, 2015.
- National Science Foundation, "DMS: Constrained Statistical Estimation and Inference: Theory, Algorithms, and Applications," $380,000, 2015.

[redacted]

- Office of Naval Research, "Statistical Machine Learning with Restrictions on Shape, Structure and Computation," $400,000, 2015.
- Amazon AWS Machine Learning in Education Research Grant, $20,000, 2014.
- Office of Naval Research, "Statistical Learning, Optimization, and Decision Making," $1,125,000, 2011–2015.
- National Science Foundation, "IIS-Small: Nonparametric Structure Learning for Complex Scientific Datasets," $170,000, 2011–2014.
- Air Force Office of Scientific Research (AFOSR), "Statistical Machine Learning for Structured and High Dimensional Data," $900,000, 2009-2014.
- Google, "Compressed and privacy-preserving prediction," $75,000, 2007.
- National Science Foundation, "Collaborative Research: Spectral Graph Theory and Applications" (co-PI), $500,000, 2007.
- National Science Foundation, "MSPA-MCS: Nonparametric Learning in High Dimensions," $500,000, 2006.
- Google, "Dynamic and structured topic models," $150,000, 2006.
- National Science Foundation, "ITR: Machine Learning of Sequences and Structured Data: Tools for Non-Experts" (co-PI), $1,000,000, 2004–2007.
- National Science Foundation, "ITR: Machine Learning from Labeled and Unlabeled Data," $395,000, 2003–2006.
- National Science Foundation, "ITR: NLP and Machine Learning Techniques for Modeling Biological Sequences" (co-PI), subcontract to UPenn, $268,989, 2001–2006.
- National Science Foundation/ARDA, "Maintenance and Development of the Lemur Language Modeling Toolkit," $505,580, 2003–2004.
- National Science Foundation, "ITR/AP: ALADDIN: A Center for ALgorithm ADaptation, Dissemination, and INtegration" (co-PI), $5,691,274, 2001–2006.

[redacted]

- National Security Agency, “Enhancement of Ad Hoc Information Retrieval using Collaborative Filtering," $570,000, 2001–2003.
- National Science Foundation, “Algorithms: From Theory to Practice" (co-PI), $600,000, 2000–2002.
- Advanced Research Development Agency (ARDA), “A Unified Framework for Information Management Using Statistical Language Models” (co-PI), $2,130,000, 2000–2003.
- National Science Foundation, “Graphical structures for coding and verification” CCR-9805366, $317,000, 1998–2001.
- IBM University Partnership Award, $80,000, 1999–2000.
- IBM University Partnership Award, $40,000, 1998–1999.
- IBM equipment grant from IBM University Relations, $4,000, 1998.
- ATR Telecommunications Research Laboratories, “Statistical natural language processing,” $250,000, 2000–2004.
- Advanced Research Development Agency (ARDA), “Language modeling for information systems” (co-PI), $1,068,498, 2000–2001.
- National Science Foundation, “Informedia-II: Integrated video information extraction and synthesis for adaptive presentation and summarization from distributed libraries” (co-PI), $4,000,000, 1999–2003.
- National Science Foundation, “Universal information access: Translingual retrieval, summarization, tracking, detection, and validation” (co-PI), $1,900,000, 1998–2001.
- National Science Foundation, “Preparation for workshop on statistical translation,” subcontract from Johns Hopkins, $50,000, 1999.
- Defense Advanced Research Projects Agency (DARPA), supplement to CMU Topic Detection and Tracking (TDT) support, $15,000, 1998.
- JustSystem research grant, $50,000, 1998–1999.

[redacted]

- Fujitsu Laboratories research grant, $25,000, 1998–1999.
- ATR Telecommunications Research Laboratories, “Statistical natural language processing,” $200,000, 1995–1999.
- Defense Advanced Research Projects Agency (DARPA), Augmentation Award for Science and Engineering Research Training (AASERT), $172,269, 1995–1998.
- National Science Foundation, “Grammatical trigrams: A new approach to statistical language modeling” (co-PI), $610,000, 1994–1997.

[redacted]

APPENDIX B

449. Below is a complete list of fields that are included in the sample I selected and shared with Prof. Chris Bail for categorization.

Table 1 (Appendix): Fields Used in Sample

#	Source	Field	Summary
[redacted]			

[363] MSFT_AI_MDL_001115484 at -488.

[364] MSFT_AI_MDL_001115484 at -490.

[365] Based on my review of [redacted] documentation and confirmed by my conversation with a Microsoft engineer, this field represents the time of a single event and can be used to sort events, as is regularly required in data science. *See*, MSFT_AI_MDL_001115484 at-488.

[366] This is equivalent to [redacted] and was created for data saving purposes.

[367] MSFT_AI_MDL_001115484 at -490.

[redacted]

#	Source	Field	Summary
[redacted]	[redacted]	[redacted]	[redacted]

[redacted]

APPENDIX C

450. As described **in Section VII.**, I conduct robustness checks to evaluate whether the calculated sample size required for the conversation review was sensitive to the choice of modeling parameters and to ensure these choices are conservative. For both the confidence interval-based, Good–Turing–based, and Chao-1-based approaches, each key parameter is systematically varied across a broad range, and the implied required sample size is computed under each configuration. The objective is to determine whether the recommended number of conversations to sample is suitable under alternative, conservative assumptions. These robustness checks are implemented in the file Robustness_Checks.ipynb and described below. Tables with all results can be found below.

A. Confidence Interval Results

Table 2 (Appendix): Confidence Interval Robustness Checks

C	p	N
20	0.25	17,139
20	0.375	21,424
20	0.5	22,852
60	0.25	20,936
60	0.375	26,170
60	0.5	27,914
100	0.25	22,717
100	0.375	28,397
100	0.5	30,290

B. Good-Turing Results

Table 3 (Appendix): Good-Turing Robustness Checks

α	k_0	k_1	p_0	N
1	1	10	0.95	93
1	1	10	0.97	126
1	1	10	0.99	195
1	1	55	0.95	426
1	1	55	0.97	547
1	1	55	0.99	893
1	1	100	0.95	723

[redacted]

α	k_0	k_1	p_0	N
1	1	100	0.97	964
1	1	100	0.99	1,577
1	6	55	0.95	304
1	6	55	0.97	384
1	6	55	0.99	592
1	6	100	0.95	557
1	6	100	0.97	705
1	6	100	0.99	1,111
1	10	55	0.95	263
1	10	55	0.97	330
1	10	55	0.99	499
1	10	100	0.95	502
1	10	100	0.97	630
1	10	100	0.99	955
1.25	1	10	0.95	112
1.25	1	10	0.97	147
1.25	1	10	0.99	278
1.25	1	55	0.95	510
1.25	1	55	0.97	698
1.25	1	55	0.99	1,235
1.25	1	100	0.95	867
1.25	1	100	0.97	1,222
1.25	1	100	0.99	2,247
1.25	6	55	0.95	339
1.25	6	55	0.97	437
1.25	6	55	0.99	689
1.25	6	100	0.95	626
1.25	6	100	0.97	825
1.25	6	100	0.99	1,355
1.25	10	55	0.95	285
1.25	10	55	0.97	362
1.25	10	55	0.99	557
1.25	10	100	0.95	551
1.25	10	100	0.97	717
1.25	10	100	0.99	1,134
1.5	1	10	0.95	128
1.5	1	10	0.97	170
1.5	1	10	0.99	296
1.5	1	55	0.95	561
1.5	1	55	0.97	844

[redacted]

α	k_0	k_1	p_0	N
1.5	1	55	0.99	1,689
1.5	1	100	0.95	898
1.5	1	100	0.97	1,435
1.5	1	100	0.99	3,135
1.5	6	55	0.95	368
1.5	6	55	0.97	489
1.5	6	55	0.99	816
1.5	6	100	0.95	704
1.5	6	100	0.97	963
1.5	6	100	0.99	1,661
1.5	10	55	0.95	305
1.5	10	55	0.97	397
1.5	10	55	0.99	622
1.5	10	100	0.95	612
1.5	10	100	0.97	810
1.5	10	100	0.99	1,343

C. Chao-1 Results

Table 4 (Appendix): Chao-1 Robustness Checks

α	k_0	k_1	p_0	N
1	1	10	0.95	30
1	1	10	0.97	39
1	1	10	0.99	87
1	1	55	0.95	176
1	1	55	0.97	219
1	1	55	0.99	321
1	1	100	0.95	329
1	1	100	0.97	366
1	1	100	0.99	697
1	6	55	0.95	88
1	6	55	0.97	124
1	6	55	0.99	192
1	6	100	0.95	187
1	6	100	0.97	259
1	6	100	0.99	398
1	10	55	0.95	79
1	10	55	0.97	97
1	10	55	0.99	144
1	10	100	0.95	170

[redacted]

α	k_0	k_1	p_0	N
1	10	100	0.97	201
1	10	100	0.99	317
1.25	1	10	0.95	38
1.25	1	10	0.97	65
1.25	1	10	0.99	119
1.25	1	55	0.95	292
1.25	1	55	0.97	414
1.25	1	55	0.99	546
1.25	1	100	0.95	598
1.25	1	100	0.97	816
1.25	1	100	0.99	1,252
1.25	6	55	0.95	110
1.25	6	55	0.97	160
1.25	6	55	0.99	208
1.25	6	100	0.95	269
1.25	6	100	0.97	369
1.25	6	100	0.99	532
1.25	10	55	0.95	96
1.25	10	55	0.97	117
1.25	10	55	0.99	201
1.25	10	100	0.95	209
1.25	10	100	0.97	278
1.25	10	100	0.99	378
1.5	1	10	0.95	66
1.5	1	10	0.97	81
1.5	1	10	0.99	162
1.5	1	55	0.95	509
1.5	1	55	0.97	728
1.5	1	55	0.99	1,087
1.5	1	100	0.95	1,315
1.5	1	100	0.97	1,624
1.5	1	100	0.99	2,637
1.5	6	55	0.95	147
1.5	6	55	0.97	197
1.5	6	55	0.99	292
1.5	6	100	0.95	397
1.5	6	100	0.97	526
1.5	6	100	0.99	775
1.5	10	55	0.95	105
1.5	10	55	0.97	146

[redacted]

α	k_0	k_1	p_0	N
1.5	10	55	0.99	234
1.5	10	100	0.95	295
1.5	10	100	0.97	375
1.5	10	100	0.99	537

[redacted]

APPENDIX D

Intentionally Omitted. [Materials-relied-upon appendices omitted per 8/20 team call; slip sheet preserves lettering.]