

Project Notes:

Project Title: The Optimization of Large Language Model (LLM) Performance with Data Preparation Techniques

Name: Armaan Priyadarshan

Note Well: There are NO SHORT-cuts to reading journal articles and taking notes from them. Comprehension is paramount. You will most likely need to read it several times, so set aside enough time in your schedule.

Contents:

Knowledge Gaps:	2
Literature Search Parameters:	3
Tags:	3
Article #1 Notes: Title	4
Article #1 Notes: Could AI-powered Robot “Companions” Combat Human Loneliness?	5
Article #2 Notes: How AI Is Transforming Genomics	8
Article #3 Notes: Art and Fun Digital Learning for Children with Special Needs	11
Article #4 Notes: Application of Artificial Intelligence in Modern Art Teaching	15
Article #5 Notes: We’re getting a better idea of AI’s true carbon footprint	19
Article #6 Notes: The Secret Water Footprint of AI Technology	22
Article #7 Notes: Energy and Policy Considerations for Deep Learning in NLP	25
Article #8 Notes: Training Compute-Optimal Large Language Models	31
Article #9 Notes: Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster	42
Article #10 Notes: Deduplicating Training Data Makes Language Models Better	54
Article #11 Notes: Beyond Scale: the Diversity Coefficient as a Data Quality Metric Demonstrates LLMs are Pre-trained on Formally Diverse Data	65
Article #12 Notes: A Pretrainer’s Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity	77
Article #13 Notes: Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets	92
Article #14 Notes: A Survey of Large Language Models	104
Article #15 Notes: Data-Juicer: A One-Stop Data Processing System for Large Language Models	117
Article #16 Notes: GLaM: Efficient Scaling of Language Models with Mixture-of-Experts	129
Article #17 Notes: Subword Regularization: Improving Neural Network Translation Models	138
Article #18 Notes: Extracting Training Data from Large Language Models	147
Article #19 Notes: The Troubling Emergence of Hallucination in Large Language Models –	–

An Extensive Definition, Quantification, and Prescriptive Remediations	160
Article #20 Notes: RoBERTa: A Robustly Optimized BERT Pretraining Approach	170
Article #21 Notes: Unified Vision and Dialogue Transformer with BERT	182
Article #22 Notes: Multi-task knowledge distillation for language model	188
Article #23 Notes: Adversarial pretraining of machine learning models	194

Knowledge Gaps:

This list provides a brief overview of the major knowledge gaps for this project, how they were resolved and where to find the information.

Knowledge Gap	Resolved By	Information is located	Date resolved

Literature Search Parameters:

These searches were performed between (Start Date of reading) and XX/XX/2024.

List of keywords and databases used during this project.

Database/search engine	Keywords	Summary of search

Tags:

Tag Name	
#ai	#llm
#greenai	#nlpalgorithms
#sustainability	#neuralnetworks
#nlp	#nlg
#datasets	#bigdata
#ml	#chatgpt
#generativeai	#efficientscaling
#opensource	

Article #1 Notes: Title

Article notes should be on separate sheets

KEEP THIS BLANK AND USE AS A TEMPLATE

Source Title	
Source citation (APA Format)	
Original URL	
Source type	
Keywords	
#Tags	
Summary of key points + notes (include methodology)	
Research Question/Problem/ Need	
Important Figures	
VOCAB: (w/definition)	
Cited references to follow up on	
Follow up Questions	

Article #1 Notes: Could AI-powered Robot “Companions” Combat Human Loneliness?

Source Title	Could AI-powered Robot “Companions” Combat Human Loneliness?
Source citation (APA Format)	Vahaba, D. (2023, July 12). <i>Could AI-powered robot “companions” combat human loneliness?</i> Duke Today. https://today.duke.edu/2023/07/could-ai-powered-robot-companions-combat-human-loneliness
Original URL	https://today.duke.edu/2023/07/could-ai-powered-robot-companions-combat-human-loneliness
Source type	Magazine
Keywords	Robots, Artificial Intelligence, Generative AI, Loneliness, Social Connection
#Tags	#robotics, #ai, #ml, #chatgpt, #mentalhealth
Summary of key points + notes (include methodology)	The article suggests that, while having a real friend is the best solution, robots might be a way for millions of socially isolated people with no other solutions to alleviate their loneliness. The article then presents an example of a social robot in the ElliQ and underscores the potential of more advanced robots embedded with modern AI, such as ChatGPT.
Research Question/Problem/Need	Can robotic companions mitigate loneliness and social isolation, especially in the elderly?
Important Figures	Nothing graphical, but statistics include: The number of Americans with no close friends has quadrupled since 1990, according to the Survey Center on American Life Serious survey of 307 care providers across Europe and the United States showed that 69% of physicians agreed that social robots could provide companionship, relieve isolation, and potentially improve patients’ mental health.
VOCAB: (w/definition)	Companion robots: robots intended for creating social connections and companionship with humans Generative AI: a form of artificial intelligence capable of producing various forms of media, including text, images, and more

Cited references to follow up on	CITATION: "Enhancing Social Connectedness With Companion Robots Employing AI," Elizabeth Broadbent, Mark Billingham, Samantha G. Boardman, P. Murali Doraiswamy. Science Robotics, July 12, 2023. DOI: 10.1126/scirobotics.ad16347
Follow up Questions	<p>How can the effectiveness of companion robots be measured?</p> <p>What are the broader societal implications of integrating companion robots to address loneliness?</p> <p>How can generative AI be made to feel more real or human, to replicate human interaction? Should it?</p>

Notes (written with assistance from ChatGPT)

- Companion robots with AI have the potential to address the loneliness epidemic, according to a report from Auckland, Duke, and Cornell Universities
- The report emphasizes the need for ethical considerations and guidelines for the development of companion robots in healthcare
- It suggests that while real human friendships are ideal, companion robots can benefit isolated individuals until social connectedness and eldercare are prioritized
- Loneliness and social isolation have serious health consequences, including mental illness, obesity, dementia, and early death
- The report highlights the potential of AI to enhance companion robots' social interaction capabilities
- Research indicates that companion robots can reduce stress and loneliness in older adults, promoting their well-being
- The lack of standardized measures to assess a robot's impact on patients underscores the need for patient-rated outcome measures
- The "Companion Robot Impact Scale" (Co-Bot-I-7) is being developed to measure the physical health and loneliness impact of companion robots
- Early results suggest that companion robots can reduce stress and promote skin healing
- Ethical guidelines are crucial for leveraging robots to create a healthier society, according to the authors

Article #2 Notes: How AI Is Transforming Genomics

Source Title	How AI Is Transforming Genomics
Source citation (APA Format)	Vacek, G. (2023, February 24). <i>How AI Is Transforming Genomics</i> . NVIDIA Blog. https://blogs.nvidia.com/blog/2023/02/24/how-ai-is-transforming-genomics/
Original URL	https://blogs.nvidia.com/blog/2023/02/24/how-ai-is-transforming-genomics/
Source type	News Article
Keywords	Whole Genome Sequencing, Accelerated Genome Analysis, Genetic Variant Discovery
#Tags	#ai, #genomics, #healthcareandlifesciences, #socialimpact
Summary of key points + notes (include methodology)	Advancements in whole genome sequencing have ignited a digital revolution in Biology, but genome data can get huge, starting at 100 gigabytes of raw data, which doubles after the use of complex algorithms and applications. Deep learning and neural networks are being used to interpret image and signal data from the billions of nucleotide pairs in the genome, and GPU-optimized and accelerated callers, such as GATK, are used to call variants, differences between the patient's sample and the reference genome. NVIDIA is enabling the next wave of genomics by powering sequencing platforms with AI base calling and variant calling.
Research Question/Problem/Need	How have AI and GPU-accelerated computing impacted the field of genomics?
Important Figures	<p>The graph illustrates two trends in genomics over time:</p> <ul style="list-style-type: none"> Cost of Sequencing (Grey Line): Shows a significant decrease from \$3 billion for the Human Genome Project in 2003 to approximately \$100 per genome in 2023. Intermediate points include \$6k per genome in 2012 and \$1k per genome in 2018. Size of Flagship Genome Projects (Green Line): Shows a significant increase from ~1 gigabyte for the First Human Genome in 2003 to 200+ petabytes for 1m+ Genomes Project(s) in 2023. Intermediate points include 12 terabytes for the 1k Genomes Project in 2012 and 21 petabytes for the 100k Genomes Project in 2018. <p><i>Increasing storage amounts and decreasing costs per genome for human genome</i></p>

	<i>data</i>
VOCAB: (w/definition)	<p>Genomics: the branch of molecular biology concerned with the structure, function, evolution, and mapping of genomes</p> <p>Deep Learning: a subset of machine learning that uses multiple layers to emulate the function of the human brain</p> <p>Base calling: the process by which an order of nucleotides in a template is inferred during a sequencing reaction</p> <p>Variant calling: the process by which variations between an individual's genome and reference genome are identified</p>
Cited references to follow up on	None
Follow up Questions	<p>How are the results from genomic sequencing and analysis with AI and GATK used?</p> <p>How exactly is AI and accelerated computing used in the process from start to finish?</p> <p>When dealing with such expensive computing and large amounts of data, what is the environmental footprint?</p>

Bulleted Notes (Written with assistance from ChatGPT)

- Advancements in whole genome sequencing are transforming digital biology
- Genomics programs are gaining momentum due to the declining cost of next-generation sequencing
- Whole genome sequencing is essential in clinical workflows and drug discovery
- Moore's law's end requires new computing approaches for efficient genome data analysis
- Sequencing a human genome generates approximately 100 gigabytes of raw data
- An estimated 40 exabytes will be needed to store all human genome data by 2025
- Deep learning and AI are improving accuracy and efficiency in genome sequencing
- Alignment algorithms like BWA-MEM and STAR are used for genomic analysis
- Variant calling is crucial for identifying genetic differences in patients and drug research
- GPU-optimized tools like GATK and DeepVariant accelerate variant calling
- NVIDIA is driving genomics advancements through AI base calling and variant calling
- Biotech companies like PacBio and Oxford Nanopore are utilizing NVIDIA technology for sequencing
- Ultima Genomics offers high-throughput whole genome sequencing at a low cost
- Singular Genomics' G4 is a powerful benchtop system for genomics research

Article #3 Notes: Art and Fun Digital Learning for Children with Special Needs

Source Title	Art and Fun Digital Learning for Children with Special Needs: A Case Study on Applying Art as a Learning Technology
Source citation (APA Format)	Bayu Tejo Sampurno, M., & Anggun Camelia, I. (2019). <i>Art and fun digital learning for children with special needs: A case study on applying art as a learning technology</i> . Proceedings of the Social Sciences, Humanities and Education Conference (SoSHEC 2019). https://doi.org/10.2991/soshec-19.2019.38
Original URL	https://www.atlantis-press.com/article/125926106
Source type	Conference Paper
Keywords	Art Education, Children with Special Needs, Pedagogical Methods, Cognitive Development
#Tags	#art, #funlearning, #artisticdevelopment
Summary of key points + notes (include methodology)	This paper aims to introduce an educational model for children based around art and technology to create a positive and effective learning environment for children with special needs. The methodology employed in this research is a qualitative case study approach, involving in-depth observations of Attention Deficit Hyperactivity Disorder children within the "Peduli Kasih Anak Berkebutuhan Khusus Surabaya" Foundation in Surabaya, Indonesia over the course of 6 months, with a focus on art education and its flexibility in enhancing the learning experiences of these children. The case study demonstrated that the educational model can provide a sense of comfort to the children, improve their development, and reduce disorders commonly experienced by them in a school setting.
Research Question/Problem/Need	How can art be used to improve learning for students with cognitive disabilities?
Important Figures	None
VOCAB: (w/definition)	Pedagogical methods: teaching methods

<p>Cited references to follow up on</p>	<p>G. R. of the R. of Indonesia, “Number 32 of 2013, concerning Amendments to Government Regulation Number 19 of 2005 concerning National Education Standards.,” 77I, paragraph 1, letter g. [Online]. Available: http://sindiker.dikti.go.id/dok/PP/PP32-2013PerubahanPP19-2005SNP.pdf. [Accessed: 16-Nov-2015].</p> <p>H. Read, <i>Education Through Art</i>. London: Faber and Faber, 1970.</p> <p>D. Kelly, <i>Uncovering the History of Children’s Drawing and Art</i>. Westport: Praeger Publishers, 2004.</p> <p>G. R. of the R. of Indonesia, “UU No.2 of 1989 concerning the National Education System.” .</p> <p>H. Thompson and F. Evans, <i>The PDA Paradox: The Highs and Lows of My Life on a Little-Known Part of the Autism Spectrum</i>. London: Jessica Kingsley Publishers, 2019.</p> <p>V. Lowenfeld and L. Brittain, <i>Creative and Mental Growth</i>. New York: Macmillan Publishing, 1982.</p> <p>C. Magnusen, <i>Teaching Children With Autism and Related Spectrum Disorder, Art and Science</i>. Florida: Alfred Publishing, 2006.</p>
<p>Follow up Questions</p>	<p>Could this study be conducted on a larger scale?</p> <p>What methods specifically would comprise the educational model?</p> <p>Could they be applied to more demographics?</p> <p>What are the potential challenges of integrating art into special education?</p>

Notes (written with assistance from ChatGPT)

Introduction

- The presence of art education in Indonesian schools is a concern for the government and education experts
- The law emphasizes that art and cultural study materials aim to shape students' character, fostering art and cultural understanding
- Art education is considered as important as other subjects, providing a balance of logical-rational and ethical-moral education
- Plato and Aristotle supported the idea that art should be the basis of education
- Education is a conscious effort to create a learning atmosphere and process where students actively develop their potential
- Art provides complex learning material for children, including those with special needs, and encourages education through play
- Art is seen as a fundamental and unifying activity that allows children to express themselves and develop sensitivity
- Art education helps in internalizing aesthetic experiences, training high sensitivity, and fostering creativity
- Education for children with special needs in Indonesia often focuses on output and neglects the outcomes and the importance of play and pleasure in education
- The paper aims to provide fun learning methods for children with special needs through the flexibility of art
- Research on art therapy primarily focuses on using art to address health and psychological issues
- Melinda J. Emery's research on "Art Therapy as an Intervention for Autism" explores the role of art in the development of autistic children and their visual communication
- Carl E. Stafstorm, Janice Haviena, and Anthony J. Krezinski examine the effects of art therapy on children and adolescents with epilepsy, noting improved cognitive benefits based on parental feedback
- Chung-Hsin Chiang, Wei-Tsuen Soong, Tzu-Ling Lin, and Sally J. Rogers study nonverbal communication skills in young children with autism through art therapy
- Annette Marjorie Miller-Jones investigates the impact of music therapy on language acquisition for children on the autism spectrum, noting progress in speech and social skills.
- Beth Nemes's research focuses on family-based music therapy, emphasizing collaborative action between music and family therapy for educational and therapeutic purposes

Methods

- The paper's goal is to introduce enjoyable learning methods for children with special needs using art's flexibility
- Existing art therapy research mainly concentrates on addressing health and psychological issues.
- Melinda J. Emery's research on "Art Therapy as an Intervention for Autism" delves into how art contributes to the development of autistic children, particularly in terms of visual

communication

- Carl E. Stafstorm, Janice Haviena, and Anthony J. Krezinski explore the impact of art therapy on children and teenagers with epilepsy, highlighting cognitive improvements reported by parents
- Chung-Hsin Chiang, Wei-Tsuen Soong, Tzu-Ling Lin, and Sally J. Rogers investigate the enhancement of nonverbal communication skills in young autistic children through art therapy
- Annette Marjorie Miller-Jones focuses on the effect of music therapy on language acquisition in children on the autism spectrum, observing advancements in speech and social skills
- Beth Nemesh's research centers on family-based music therapy, emphasizing the collaborative use of music and family therapy for both educational and therapeutic purposes

Results and Discussion

- Art offers multiple perspectives, no complete discipline, and endless possibilities
- Art encourages people to see and hear beyond the surface of reality, making it a valuable component of a humanistic curriculum
- Art education aligns with Aristotle's concept of knowledge formation through sensation and abstraction
- Children's art education involves engaging in art activities, sensing, and eventually understanding concepts
- Learning from children with special needs, who express themselves through art, provides unique perspectives and unlimited imagination
- Artistic experiences are divided into act of production (artistic experience) and aesthetic and perception experience
- Fun learning for children with special needs requires understanding their individual needs, creating a comfortable and interactive learning environment
- Children may have different ways of being smart through various forms of expression
- Art, as a source of understanding, plays a valuable role in fun learning
- Art teachers can contribute to cross-disciplinary intelligence development
- Expertise in art education goes beyond artistic skills and involves using art as a therapeutic medium
- Interdisciplinary approaches in art education facilitate collaboration between various fields and open up new possibilities

Conclusion

- Art-based education offers comfort and enjoyment for children with special needs
- It positively impacts their development and reduces school-related disorders
- Tailoring methods to individual needs through trial and error promotes enjoyable learning
- This approach helps unlock the potential of children with special needs
- The research benefits special education teachers by highlighting the effectiveness of art-based teaching

Article #4 Notes: Application of Artificial Intelligence in Modern Art Teaching

Source Title	Application of Artificial Intelligence in Modern Art Teaching
Source citation (APA Format)	Kong, F. (2020). <i>Application of artificial intelligence in modern art teaching</i> . International Journal of Emerging Technologies in Learning (IJET), 15(13), 238. https://doi.org/10.3991/ijet.v15i13.15351
Original URL	https://www.researchgate.net/publication/342849745_Application_of_Artificial_Intelligence_in_Modern_Art_Teaching
Source type	Journal Article
Keywords	Art Teaching, Artificial Intelligence, Higher Education, Analytical Hierarchy Process, Gray Clustering
#Tags	#ai, #art, #education
Summary of key points + notes (include methodology)	The rapid development of AI has led to its gradual application in the field of higher education, an important part of which being art teaching, where AI can be used to cover shortcomings in the transfer of professional skills and knowledge. While there were a couple of prior studies on the implementation details of AI in art teaching, they still need to account for the systematic planning of AI in art teaching. Through inductive analysis, this paper analyzes the application of AI in modern art education from the two aspects of strategy analysis and model construction. It covers six main topics: prior studies on the competitiveness of higher education, the current application of AI in art teaching, the promotive role of AI in art teaching, the strategies of AI applications in modern art teaching, an application performance analysis, and a conclusion. After discussing AI's benefits and potential applications in AI art teaching, the paper constructed a performance model for application performance analysis with grey clustering.
Research Question/Problem/Need	How can AI be applied effectively in art teaching?
Important Figures	None
VOCAB: (w/definition)	Analytical hierarchy process: a method for organizing and analyzing complex decisions, using math and psychology Grey clustering: a method developed for classifying observation indices or

	observation objects into definable classes using grey incidence matrices or grey possibility functions
Cited references to follow up on	Tang, Y., He, S.Y. (2019). Research on the education mode of "artificial intelligence + higher vocational art major" entrepreneurship and employment. China Journal of Commerce, 2019(14): 223-225. https://doi.org/10.19699/j.cnki.issn2096-0298.2019.14.223 .
Follow up Questions	<p>The paper was written in 2020, focusing on less recent AI technologies, such as sentiment analysis, semantic analysis, and emotion perception. However, how could generative AI, including diffusion and large language models, play a role in art education?</p> <p>How would a personalized learning model based on the factors mentioned be trained?</p> <p>Are there specific examples of where and how these strategies were employed?</p> <p>Is there any more specific information on how the weight judgment matrix of performance indicators A was constructed?</p>

Notes (written with assistance from ChatGPT)

Introduction

- AI is increasingly applied in engineering and higher education, enhancing design and teaching
- Art education in higher education involves transmitting professional knowledge
- Some research on AI in education exists but lacks comprehensive planning for art teaching
- This paper analyzes AI's application in modern art education, discussing strategies and outcomes

Current Status of AI Application in Art Teaching

- Traditional electronic equipment like recorders and projectors are commonly used in art education
- AI aims to present art knowledge in a more intuitive way but faces a shortage of AI hardware facilities, hindering desired teaching goals
- Computer Aided Instruction (CAI) uses computer technologies to enhance art teaching
- Traditional CAI has limitations in understanding individual student needs and participation
- Stronger technical support through AI and multimedia technology is needed for modern teaching
- Current AI-based art teaching relies on Internet technology and online platforms, offering innovative learning options
- However, it often lacks the artistic teaching atmosphere, especially for large groups
- The disconnect between modern art teaching concepts and intelligent teaching modes is a challenge
- AI technology is not yet fully developed, limiting its potential in art education
- AI struggles with tasks like sentiment analysis and subjective aesthetic evaluation
- Current AI-based art teaching is basic and doesn't fully utilize AI's capabilities, highlighting the need to overcome AI and art education integration challenges

The Promotive Role of AI in Art Teaching

- The application of AI in art teaching aims to improve the learning effect of art students by simulating human thinking processes, enhancing logical thinking, and personalizing education
- AI-based art teaching systems can adapt to students' individual needs, provide personalized resources and learning paths, and facilitate teacher-student interactions
- AI can enrich the teaching methods of art teachers, allowing them to focus on innovative teaching activities and one-on-one problem discovery, with AI assisting in various teaching tasks
- AI can improve the art teaching environment by optimizing visual and auditory presentations, spatial layout, and environmental factors, creating a better user experience for teachers and students
- AI can enhance art teaching methods by using advanced digital media and VR, providing students with a more immersive and multisensory art experience
- AI enables personalized and procedural art teaching evaluation, recording students' interactions and learning behaviors to provide a comprehensive understanding of their progress and

encourage self-improvement

Strategies for the Implementation of AI in Modern Art Teaching

- The role of AI in college art teaching is growing, necessitating a change in the role of art teachers to leverage AI's capabilities for personalized and objective education
- AI can enhance personalization in art teaching, providing personalized content and intelligent analysis, balancing scale and customization, and improving the learning experience
- AI can help collect and analyze art education data, support management services, and improve decision-making and resource allocation in art education
- The application of AI in art teaching requires the expansion of art education data, knowledge graph technology, and standard data systems
- AI can transform the art teaching environment, creating a digitalized and intelligent teaching atmosphere, and improving personalized service levels
- The integration of online and offline classes and the use of VR can provide immersive learning experiences and enhanced teaching effectiveness in art education

Application Performance Analysis of AI in Modern Art Teaching

- The evaluation of AI application in modern art teaching should be guided by scientific, objective, and targeted principles
- Performance indicators, including art teaching mode, art teaching method, art teaching content, teaching atmosphere, teaching means, teaching effect, and teaching environment, are selected for analysis
- The Analytic Hierarchy Process (AHP) method is used to assign weights to these performance indicators
- The application performance is divided into different degrees
- Gray clustering functions are used to analyze the performance of each indicator
- A weighted gray correlation degree is calculated to assess the performance of AI application for a given object
- The performance degree of an object is determined based on its correlation with the predefined performance degrees

Conclusion

- The paper analyzed the application of AI in college art teaching
- It discussed both the shortcomings and the positive impact of AI in college art teaching
- The paper proposed strategies for using AI to support art teaching
- To measure the application effect of AI, a performance model was constructed
- The study combines theoretical analysis and calculation models to analyze the application strategies of AI in modern art teaching, offering innovation in theory and practical engineering applications

Article #5 Notes: We're getting a better idea of AI's true carbon footprint

Source Title	We're getting a better idea of AI's true carbon footprint
Source citation (APA Format)	Heikkil, M. (2022, November 15). <i>We're getting a better idea of AI's true carbon footprint</i> . MIT Technology Review. https://www.technologyreview.com/2022/11/14/1063192/were-getting-a-better-idea-of-ais-true-carbon-footprint/
Original URL	https://www.technologyreview.com/2022/11/14/1063192/were-getting-a-better-idea-of-ais-true-carbon-footprint/
Source type	News Article
Keywords	Artificial Intelligence, Large Language Models, Sustainability, LLM Optimization
#Tags	#ai, #greenai, #greensoftware, #llm
Summary of key points + notes (include methodology)	Large language models, a rapidly growing form of artificial intelligence, require abundant energy to train and can leave a significant environmental impact. The article explores AI startup Hugging Face's paper and delves into their method of more precisely calculating the carbon footprint of these models. Hugging Face demonstrated its approach by measuring the emissions of BLOOM, its own large language model, throughout its entire lifecycle, factoring in emissions beyond training, such as those produced by the manufacturing of the computer equipment used for training and the broader computing infrastructure. While BLOOM primarily uses nuclear energy, other LLMs trained worldwide use fossil fuels, which could be even more polluting. The paper also calls attention to the possible inaccuracies of emission calculations of other popular LLMs, such as GPT-3 and Meta's OPT, calling for a more thorough evaluation of the environmental impact of LLMs and AI in general. It also sheds light on the scale of the carbon footprints, which is important for companies and developers to know as they balance the trade-offs between costs, efficiency, and sustainability. This article would serve as a solid foundation for developing a methodology for investigating the emissions of various LLMs throughout their lifetimes to explore strategies of optimization and mitigation, such as alternative training techniques and diverse energy sources.
Research Question/Problem/Need	How can the environmental impact of training large language models be gauged throughout the entirety of the training lifecycle?

Important Figures	None
VOCAB: (w/definition)	<p>LLMs (Large Language Models): A type of artificial intelligence that has been trained on vast amounts of data to understand and generate text</p> <p>CodeCarbon: A software tool used for tracking the carbon dioxide emissions produced by running AI models in real-time</p>
Cited references to follow up on	None
Follow up Questions	<p>How can AI developers and researchers effectively reduce the carbon footprint of large language models during their entire life cycle, considering factors like training, hardware manufacturing, and ongoing operation?</p> <p>What methodologies and standards can be established to measure and report carbon emissions accurately for AI models, ensuring transparency and comparability across different AI projects and organizations?</p> <p>In the context of AI research, what strategies and practices can be adopted to fine-tune existing models for specific tasks without significantly increasing their energy consumption and environmental impact, and how do these compare to developing larger models from scratch?</p> <p>Generated with ChatGPT</p>

Notes (written with assistance from ChatGPT)

- Hugging Face, an AI startup, aims to provide a more precise calculation of the carbon footprint of large language models (LLMs) by considering the emissions throughout an LLM's entire life cycle
- This approach involves estimating emissions during training and the ongoing usage of LLMs, which could provide more realistic data about the environmental impact of AI products
- Hugging Face tested its approach on its own large language model, BLOOM, and found that its carbon emissions doubled when accounting for the manufacturing of computer equipment, computing infrastructure, and ongoing energy use
- The results vary depending on where the LLM is trained, with models in regions powered by nuclear energy being less polluting than those in areas with fossil fuel-reliant energy grids
- Comparatively, models like GPT-3 and Meta's OPT were estimated to emit significantly higher levels of carbon emissions
- Hugging Face's research sets a new standard for measuring the carbon footprint of AI models and emphasizes the importance of understanding the environmental impact of large language models
- The findings encourage companies and developers to make choices that limit the carbon footprint of AI systems, possibly by focusing on more efficient ways of conducting AI research and fine-tuning existing models
- The impacts of AI on the environment are not inevitable, and the choices made about AI algorithms and their usage play a crucial role in addressing carbon emissions

Article #6 Notes: The Secret Water Footprint of AI Technology

Source Title	The Secret Water Footprint of AI Technology
Source citation (APA Format)	Syed, N. (2023, April 15). <i>The secret water footprint of AI technology</i> . The Markup. https://themarkup.org/hello-world/2023/04/15/the-secret-water-footprint-of-ai-technology
Original URL	https://themarkup.org/hello-world/2023/04/15/the-secret-water-footprint-of-ai-technology
Source type	News Article
Keywords	Artificial Intelligence, Water Footprint, Environment, Sustainability, Dynamic Scheduling
#Tags	#ai, #sustainability, #greenai, #water
Summary of key points + notes (include methodology)	This article explores AI models' water footprint or consumption and the environmental ramifications. A study conducted at the University of Texas at Arlington discovered that training GPT-3, a vast large language model, in Microsoft's modern data centers in the United States directly consumed 700,000 liters of freshwater, and other data centers in different geographical locations might've consumed up to triple that amount. The article also dove into other concerns of water efficiency, notably the fact that high water efficiency doesn't necessarily correlate to high carbon efficiency, and they can often conflict, showing that many factors must be considered when developing truly sustainable AI. Furthermore, the cooling of data centers can have a significant environmental impact on their locality. The study showed that the timing and location of large AI model training significantly impact the water footprint due to the spatial-temporal diversity of water usage effectiveness, and dynamic scheduling can reduce water consumption. Transparency regarding the water footprints of AI models was emphasized as it would facilitate measurement, benchmarking, and improvement, allowing developers to maximize efficiency with dynamic scheduling and helping users better know their water footprint. This article is an excellent foundation for research on measuring the environmental impact of AI as a whole across various geo-temporal circumstances for developing optimization strategies.

Research Question/Problem/Need	What is the water footprint of training AI and how can it be mitigated?
Important Figures	<p style="text-align: center;">700,000</p> <p style="text-align: center;">Number of liters of clean freshwater to train GPT-3 in Microsoft's U.S. data centers, not including electricity generation</p> <p style="text-align: center;"><i>The immense amount of water needed to train GPT-3, a popular language model</i></p>
VOCAB: (w/definition)	<p>Water footprint: An environmental indicator that measures the volume of fresh water needed to produce something</p> <p>Data center: A data center is a physical location that stores computing machines and their related hardware equipment</p> <p>Spatial-temporal diversity: The variation or differences in both space (spatial) and time (temporal) aspects within the context of training AI models</p>
Cited references to follow up on	None
Follow up Questions	<p>How can transparency regarding the water footprint of AI models lead to more sustainable practices, and what steps can AI developers take to reduce water consumption while maintaining performance?</p> <p>What are the potential challenges and trade-offs between reducing the carbon footprint and conserving water in the development and operation of AI technologies?</p> <p>Given the spatial-temporal variations in water efficiency, what computational techniques or algorithms can be employed to dynamically schedule AI workloads and tasks to minimize the water footprint while maintaining performance metrics?</p> <p>Generated with ChatGPT</p>

Notes (written with assistance from ChatGPT)

- Research by the University of California, Riverside focuses on the water footprint of AI technology, which has received less attention compared to its carbon footprint
- Large-scale AI models, like GPT-3, are found to be significant water consumers, with Microsoft's GPT-3 training using 700,000 liters of clean freshwater in US data centers, not including electricity generation
- For AI inference, a ChatGPT conversation consumes approximately 500 ml of water for a short interaction, varying with usage time and location
- The unique spatial-temporal diversities of AI models' runtime water efficiency indicate potential reductions in water consumption by scheduling AI workloads at specific times and locations
- This study highlights the importance of incorporating water considerations into AI development
- Water-conscious users may opt for water-efficient AI model usage times and data centers
- Transparency is proposed as a next step to measure, benchmark, and improve AI models' water footprint, helping developers schedule training and inference efficiently and inform users of their water consumption
- Transparency can lead to better water management and efficient use of AI models, similar to Apple's clean energy scheduling for charging devices

Article #7 Notes: Energy and Policy Considerations for Deep Learning in NLP

Source Title	Energy and Policy Considerations for Deep Learning in NLP
Source citation (APA Format)	Strubell, E., Ganesh, A., & McCallum, A. (2019). <i>Energy and policy considerations for deep learning in NLP</i> (arXiv:1906.02243). arXiv. https://arxiv.org/abs/1906.02243
Original URL	https://arxiv.org/abs/1906.02243
Source type	Journal Article
Keywords	Machine Learning, Deep Learning, NLP, Sustainability, Green AI, Cost Analysis, Hyperparameter Optimization
#Tags	#ai, #neuralnetworks, #greensoftware, #nlpalgorithms
Summary of key points + notes (include methodology)	While advances in techniques and hardware for training deep neural networks have led to significant improvements in accuracy across NLP tasks, training a state-of-the-art model on abundant amounts of data can have a significant financial and environmental cost. This paper aims to quantify and approximate the said costs based on data from recently trained NLP neural network models. To characterize the dollar cost and carbon emissions, the researchers estimated the kilowatts of energy required to train various popular NLP models, including the Transformer, ELMo, BERT, and GPT-2, and converted them to approximate carbon emissions and electricity costs. Additionally, they also estimated the resources required to transfer an existing model to a new task or develop new models by performing a case study on the computational resources required to tune LISA, a state-of-the-art NLP model. They stress the importance of researchers disclosing training durations and how models respond to hyperparameter adjustments. This disclosure aids in making model comparisons and gauging their compatibility with available resources. Additionally, the article advocates for providing academic researchers with fair access to computing resources while also urging the advancement of NLP research through the creation of more energy-efficient algorithms and hardware.
Research Question/Problem/Need	What are the environmental and financial costs of training NLP models and how can they be mitigated?

Important Figures

Consumption	CO ₂ e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO₂ emissions from training common NLP models, compared to familiar consumption.¹

The estimated carbon emissions of common NLP model training compared to other tasks such as a flight or car usage over a lifetime

Consumer	Renew.	Gas	Coal	Nuc.
China	22%	3%	65%	4%
Germany	40%	7%	38%	13%
United States	17%	35%	27%	19%
Amazon-AWS	17%	24%	30%	26%
Google	56%	14%	15%	10%
Microsoft	32%	23%	31%	10%

Percent distribution of energy sources among cloud compute providers

Model	Hardware	Power (W)	Hours	kWh-PUE	CO ₂ e	Cloud compute cost
Transformer _{base}	P100x8	1415.78	12	27	26	\$41–\$140
Transformer _{big}	P100x8	1515.43	84	201	192	\$289–\$981
ELMo	P100x3	517.66	336	275	262	\$433–\$1472
BERT _{base}	V100x64	12,041.51	79	1507	1438	\$3751–\$12,571
BERT _{base}	TPUv2x16	—	96	—	—	\$2074–\$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	\$942,973–\$3,201,722
NAS	TPUv2x1	—	32,623	—	—	\$44,055–\$146,848
GPT-2	TPUv3x32	—	168	—	—	\$12,902–\$43,008

Estimated cost of training models in terms of CO₂ emissions (lbs) and cloud compute cost (USD)

Models	Hours	Estimated cost (USD)	
		Cloud compute	Electricity
1	120	\$52–\$175	\$5
24	2880	\$1238–\$4205	\$118
4789	239,942	\$103k–\$350k	\$9870

Estimated cost in terms of cloud compute and electricity for training: (1) a single model (2) a single tune and (3) all models trained during R&D

VOCAB: (w/definition)

NLP (Natural Language Processing): A machine learning technology that gives computers the ability to comprehend, interpret, and manipulate linguistic data

	<p>Neural architecture search: The automated process of finding the optimal neural network architecture for a specific machine learning task</p> <p>Hyperparameters: Parameters whose values control the learning process and determine the values of model</p> <p>Power Usage Effectiveness: A metric used to determine the energy efficiency of a data center</p> <p>BLEU Score: A metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations.</p> <p>Gigaflops: A measure of computing performance, specifically one billion floating-point operations per second. It is used to quantify the computational power required for training and running neural network models.</p>
Cited references to follow up on	<p>Da Li, Xinbo Chen, Michela Becchi, and Ziliang Zong. 2016. Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. <i>2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)</i>, pages 477–484.</p> <p>James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. <i>Journal of Machine Learning Research</i>, 13(Feb):281–305.</p>
Follow up Questions	<p>What is meant by “training off-the-shelf” models for one day?</p> <p>What are the potential advancements in hardware and software technologies that could lead to more sustainable and accessible NLP research?</p> <p>How can hyperparameter tuning, alternative architectures, and hardware optimization mitigate the environmental costs of NLP training?</p> <p>What does the availability of renewable energy sources for data centers look like across different geographic locations?</p> <p>How can this methodology be adjusted to account for the entirety of the training lifecycle?</p> <p>What are carbon emissions like for other types of AI models?</p>

- Developments in the capabilities of natural language processing models have been significant, but have also increased model sizes and computational requirements
- Research and experimentation with hyperparameters exacerbates the environmental and financial costs of training
- As highlighted in the graph, training newer models, such as the Transformer, takes much more energy than previous NLP models
 - Additionally, the experimentation aspects and neural architecture search, played an extreme role on the total carbon footprint
 - Training the Transformer with neural architecture search emitted 5 times as much carbon as the use of a car in a whole lifetime
- Many NLP models now require specialized hardware to train, and there's a significant environmental cost associated with leaving it running for hours or weeks at a time
- Generally, the energy used to train is non-renewable because not all locations have renewable energy, and even if they did, most of it is not being allocated to model training
- The researchers aim to estimate the kilowatts of energy needed to train NLP models and the resources required to tune existing models
- The conclusions outlined are that:
 - Time to retrain and sensitivity to hyperparameters should be reported for NLP machine learning models
 - Academic researchers need equitable access to computational resources
 - Researchers should prioritize developing efficient models and hardware

Methods

- To measure the energy required for training, they trained off-the-shelf models using the default settings and sample GPU and CPU power consumption during training
- They estimated the total time needed to train to completion using training times and hardware from the original papers
- They calculated the total power required using the CPU, DRAM, and GPU power consumption as well as the PUE
- From there, they converted the power to estimated carbon emissions
- 2.1 Models
 - The models analyzed include the Transformer, ELMo, BERT, and GPT-2
 - Details on the specifics of each model were covered

Related Work

- The author mentioned related work for models in computer vision, such as convolutional neural networks for classification
 - This work measures average power draw required during inference on GPUs as a function of the batch size
 - It doesn't analyze recurrent or self-attention models, however
- During the time of writing, there wasn't an analysis of the computation for R&D and hyperparameter tuning in NLP

Experimental Results

Model	Hardware	Power (W)	Hours	kWh-PUE	CO ₂ e	Cloud compute cost
Transformer _{base}	P100x8	1415.78	12	27	26	\$41–\$140
Transformer _{big}	P100x8	1515.43	84	201	192	\$289–\$981
ELMo	P100x3	517.66	336	275	262	\$433–\$1472
BERT _{base}	V100x64	12,041.51	79	1507	1438	\$3751–\$12,571
BERT _{base}	TPUv2x16	—	96	—	—	\$2074–\$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	\$942,973–\$3,201,722
NAS	TPUv2x1	—	32,623	—	—	\$44,055–\$146,848
GPT-2	TPUv3x32	—	168	—	—	\$12,902–\$43,008

- TPUs were more cost efficient than GPUs on workloads that make sense for the hardware
- NAS cost \$150,000 in on-demand compute time non-trivial carbon emissions for marginal improvement in its BLEU score
- 4.2 Cost of development: Case study
 - For evaluating R&D costs, the researchers looked into the logs of training required to develop LISA, a multi-task NLP model
 - LISA served as a representative NLP pipeline
 - The results indicated that while training a new model isn't too expensive financially or environmentally, tuning and developing one for a new dataset can become extremely expensive

Conclusions

- Authors should report training time and sensitivity to hyperparameters
 - It would be beneficial to compare different model architectures and perform cost-benefit analysis
 - Authors of models that are going to be fine-tuned for new tasks later on should report training time and the computational resources required
 - For this to happen, they need
 - A standard, hardware-independent measurement of training time, such as gigaflops required to convergence
 - A standard measurement of model sensitivity to data and hyperparameters, such as variance with respect to hyperparameters searched
- Academic researchers need equitable access to computation resources
 - The costs associated with computational requirements are not accessible to everyone
 - The models trained in this paper required industry access to large-scale compute, but limiting NLP research to industry labs hurts it
 - The cost of building in-house resources causes people to rely on cloud compute services, such as AWS, Azure, and GCP
 - It is often more cost effective for researchers to pool resources and build shared compute centers
 - A government-funded academic compute cloud would provide equitable access to all researchers
- Researchers should prioritize computationally efficient hardware and algorithms
 - Promote research for more energy-efficient algorithms and hardware in NLP.

- Encourage NLP software developers to prioritize efficient models
- Advocate the use of efficient hyperparameter tuning techniques like random or Bayesian search
- Integrate these techniques into familiar NLP workflows for reduced energy consumption

Article #8 Notes: Training Compute-Optimal Large Language Models

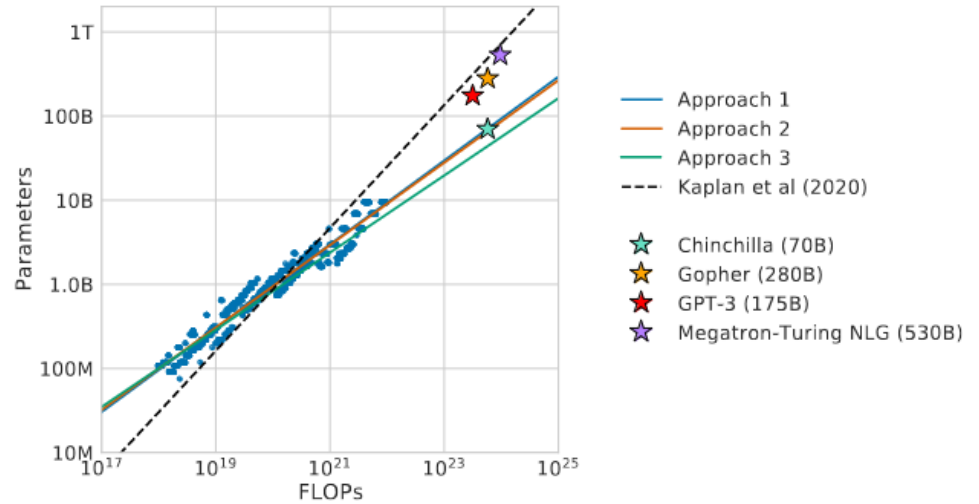
Source Title	Training Compute-Optimal Large Language Models
Source citation (APA Format)	Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. de L., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G. van den, Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., ... Sifre, L. (2022). <i>Training compute-optimal large language models</i> (arXiv:2203.15556). arXiv. https://arxiv.org/abs/2203.15556
Original URL	https://arxiv.org/abs/2203.15556
Source type	Journal Article
Keywords	Artificial Intelligence, Generative AI, Large Language Models, Transformers, Model Scaling, DeepMind, Chinchilla, Gopher
#Tags	#ai, #languagemodels, #generativeai, #deepmind
Summary of key points + notes (include methodology)	Large language models have grown significantly in size, but often remain undertrained due to improper scaling. This paper explored how optimal model performance can be achieved under a given compute budget and the corresponding allocation of parameters and tokens. It explored how model performance can be scaled efficiently in lieu of solely increasing model size while keeping the dataset constant. A significant benefit of training compute-optimal models is reduced compute costs for fine-tuning and inference. The researchers trained over 400 language models with varying sizes and tokens and identified the importance of scaling the number of tokens with parameters. From there, they tested their hypothesis by training and benchmarking Chinchilla, a compute-optimal LLM, with the same compute budget as Gopher, a 280-billion parameter model trained by Deepmind, with a quarter of the parameters and significantly more tokens. The article starts by and continues citing the results of research by Jared Kaplan and others that showed a power law relationship between language models and model size, dataset size, compute amounts for training, and other architectural details. From there, it explores three different approaches to analyzing the relationship between model size and the number of tokens under a given compute budget. It tries to fit an empirical estimator using

training curves from various models to determine optimal scaling for maximizing performance and minimizing loss. Using this empirical estimator, they trained and extensively evaluated Chinchilla.

Research Question/Problem/Need

Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?

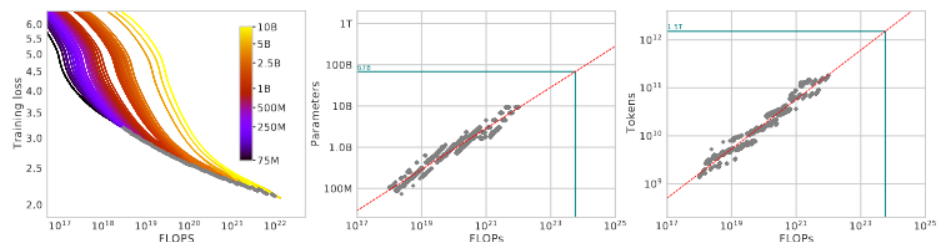
Important Figures



Overlaid projections from the approaches presented in the paper and Kaplan’s research that demonstrate how other LLMs are much larger than they need to be

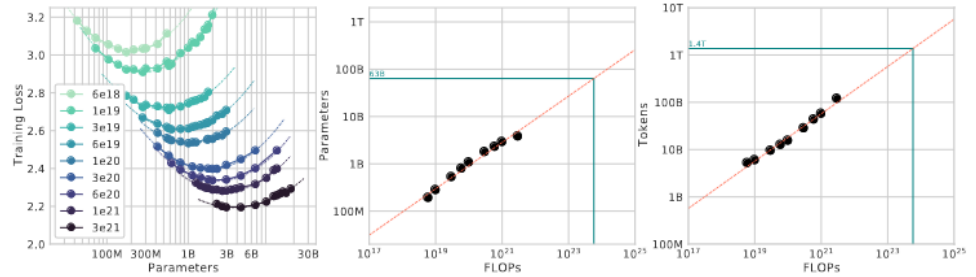
Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
<i>Chinchilla</i>	70 Billion	1.4 Trillion

A table illustrating the size and number of tokens of various other LLMs and Chinchilla, highlighting the disparity in training tokens and how other models don’t have enough

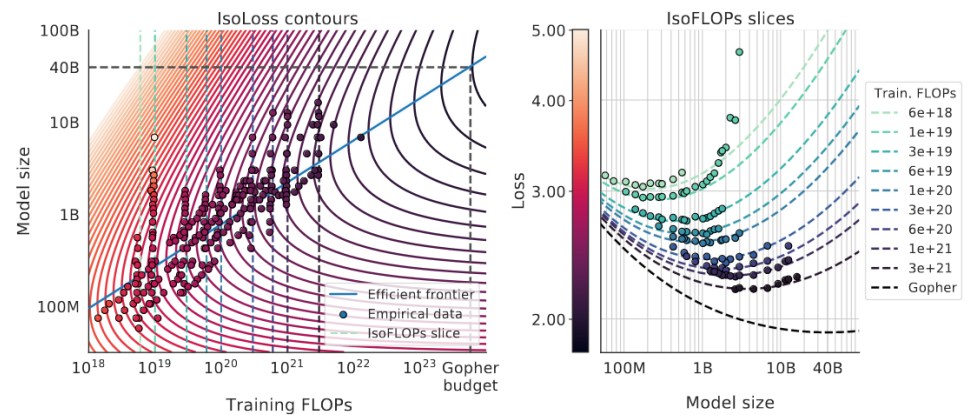


The left graph illustrates all the different runs of training models ranging from 70M

to 10B parameters. The learning curves, especially the points with minimum loss per FLOP, were used to find linear fits for approximating the optimal size (center) and number of tokens (right) for a given FLOP budget.



The graph on the left shows the relationship between different numbers of tokens and model size with a constant amount of final FLOPs. There are clear valleys in loss, showing that there is an optimal model to train. The optimal model sizes and number of tokens are projected using these valleys in the graphs in the center and right.



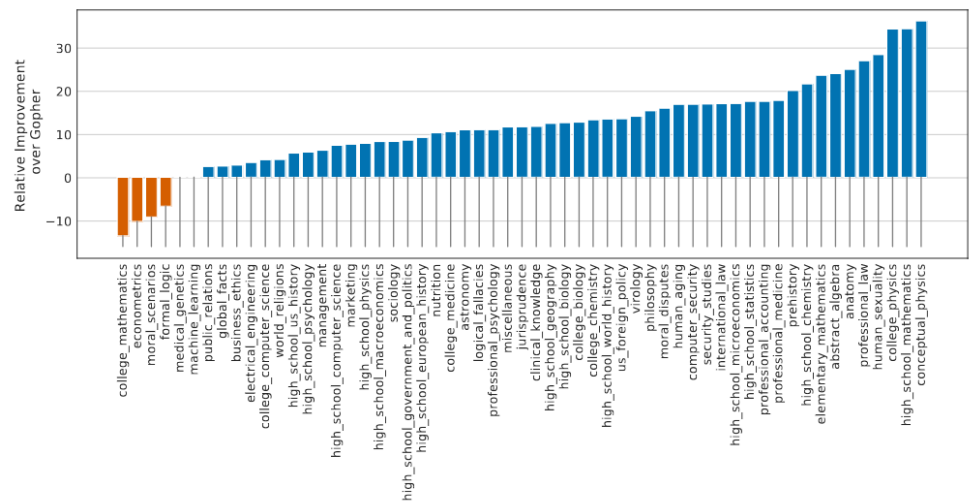
A fit of the parametric modeling of the loss, display contour (left), and isoFLOP slices (right).

Parameters	FLOPs	FLOPs (in <i>Gopher</i> unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Estimated optimal training FLOPs and training tokens for various model sizes.

Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
<i>Gopher</i> 5-shot	60.0%
<i>Chinchilla</i> 5-shot	67.6%
Average human expert performance	89.8%
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

Massive Multitask Language Understanding (MMLU). The average 5-shot accuracy over 57 tasks with model and human accuracy comparisons taken from Hendrycks et al. (2020).



MMLU results compared to Gopher.

VOCAB: (w/definition)

Power law relationship: A functional relationship between two quantities, where a relative change in one quantity results in a relative change in the other quantity proportional to a power of the change, independent of the initial size of those quantities

FLOPs: Floating Operations Per Second; a measure of computing performance

Loss: A mathematical function that quantifies the difference between predicted and actual values in a machine learning model

Parameters: Numerical values that define the behavior of LLMs and neural networks and contain the weights and biases

Tokens: Basic units of text that LLMs use to represent characters, words, subwords, or other segments of text or code using numbers

Learning rate: A hyperparameter used to govern the pace at which an algorithm updates or learns the values of a parameter estimate

Batch size: The number of samples processed before the model is updated

Cited references to follow up on

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf>.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Follow up Questions

How would training compute-optimal models affect carbon emissions?

Is there a quantifiable improvement in sustainability during finetuning and inference?

Would the increase in tokens needed be an issue? (data leakage and just gathering such immense amounts of data)

How does one scale a dataset?

How would the data be quality tested?

Does training with a larger dataset cause more carbon emissions during training?

Was there a reason Chinchilla did worse in certain tasks than Gopher?

Is a linear fit most appropriate for modeling optimal model sizes and the number of tokens?

Notes (written with assistance from ChatGPT)

Introduction

- Introduction of Large Language Models (LLMs) with over 500 billion parameters and impressive performance
- Acknowledgment of substantial compute and energy costs for training large language models
- Emphasis on the need to accurately estimate model hyperparameters for a given compute budget
- Reference to the power law relationship between model parameters and performance
- Suggestion that large models should be trained for more training tokens than previously recommended
- The common practice of training large models for approximately 300 billion tokens
- The central research question: How should one balance model size and the number of training tokens within a fixed computational budget
- The optimization goal of minimizing pre-training loss under the constraint of a fixed FLOPs budget
- Empirical estimation of optimal allocation functions based on over 400 models with varying parameters and training horizons
- Prediction that a more compute-optimal model, Chinchilla, should be smaller and trained on more tokens, leading to improved performance and reduced inference cost
- A reference to the energy cost of large language models and their benefits beyond immediate performance improvements
- Table 1 showing details of current large dense transformer models and their training tokens, introducing Chinchilla as a smaller model trained for an extended period

Related Work

- Acknowledgment of challenges in LLMs, including computational requirements and the need for high-quality training data
- Importance of understanding scaling behavior and transfer properties in LLM development
- Reference to Kaplan et al. (2020) showing a predictable relationship between model size and loss
- Differences between the current analysis and Kaplan et al. (2020), including variable hyperparameters and model sizes up to 16B parameters
- Mention of Clark et al. (2022) investigating scaling properties of Mixture of Expert language models
- Consideration of hyperparameters beyond model size and training tokens, including learning rate, batch size, and depth-to-width ratio
- Introduction of alternative model architectures such as conditional computation MoE models and models with explicit retrieval mechanisms

Estimating the optimal parameter/training tokens allocations

- Three approaches to address the research question: How to balance model size and training tokens within a fixed FLOPs budget

- Training a range of models with varying sizes and training tokens.
- Assuming a power-law relationship between compute and model size
- Similar predictions from all three methods, indicating that parameter count and training tokens should increase proportionally with more compute

3.1. Approach 1: Fix model sizes and vary number of training tokens

- The first approach involves varying the number of training steps for a fixed set of models ranging from 70M to over 10B parameters
- This approach helps estimate the minimum loss achieved for a given number of training FLOPs
- It includes details on training steps and interpolation of training loss curves
- The results are used to create mappings from FLOP count to optimal model size and number of training tokens for any given amount of compute
- Power laws are fitted to estimate these mappings, resulting in values of $a = 0.50$ and $b = 0.50$, as summarized in Table 2
- A head-to-head comparison at 1021 FLOPs demonstrates the advantage of the predicted model size over the approach of Kaplan et al. (2020)

3.2. Approach 2: IsoFLOP profiles

- In the second approach, the model size is varied for a fixed set of 9 different training FLOP counts, ranging from 6×10^{18} to 3×10^{21} FLOPs
- The focus is on determining the final training loss for each combination of model size and FLOP budget
- Parabolas are fitted to the IsoFLOPs curves to identify the model size at which the minimum loss occurs for each FLOP budget
- Power laws are then fitted to establish the relationship between FLOPs, loss-optimal model size, and the number of training tokens
- The resulting exponents a and b are summarized in Table 2

3.3. Approach 3: Fitting a parametric loss function

- A parametric function is introduced to model the final losses from experiments in Approach 1 and 2, aiming to capture the relationship between model parameter count and the number of seen tokens.
- The parametric function comprises three terms: the loss for an ideal generative process, the performance of a perfectly trained transformer, and the effect of not training to convergence
- Parameters of the function (A, B, E, α, β) are estimated by minimizing the Huber loss between predicted and observed log loss using the L-BFGS algorithm

- The choice of Huber loss ensures robustness against outliers in the data
- Efficient computational frontiers are constructed by minimizing the parametric loss while constraining FLOPs to scale with model size and data
- Figure 4 displays contour plots and isoFLOP slices that illustrate the fitted parametric loss function. It also includes the efficient computational frontier represented by a blue line in log-log space
- From this analysis, specific values for parameters a and b in the power-law form of the efficient computational frontier are derived, with $a = 0.46$ and $b = 0.54$, as summarized in Table 2

3.4. Optimal model scaling

- The three approaches used in the study, despite employing different fitting methodologies and models, consistently suggest that as the computational budget increases, model size and the amount of training data should be scaled in roughly equal proportions
- Approach 1 and Approach 2 produce very similar predictions for optimal model sizes, while Approach 3 predicts even smaller models to be optimal at larger compute budgets
- It's noted that points with lower training FLOPs have larger residuals, which can be attributed to the empirically observed negative curvature in the optimal scaling frontier
- Table 3 presents estimated numbers of FLOPs and tokens required to train compute-optimal models of various sizes, highlighting that many current large language models are considerably over-sized given their compute budgets
- For example, a 175 billion parameter model should be trained with approximately 4.41×10^{24} FLOPs and over 4.2 trillion tokens, and a 280 billion parameter model is optimal with approximately 10^{25} FLOPs and 6.8 trillion tokens
- The analysis underscores the importance of dataset collection in addition to engineering improvements to achieve optimal model performance
- Further analyses on additional datasets (C4 and GitHub code) confirm the conclusion that model size and the number of training tokens should be scaled proportionally

Chinchilla

- Based on the analysis, the optimal model size for the Gopher compute budget falls between 40 and 70 billion parameters
- To test this hypothesis, the study trained a model with 70 billion parameters, referred to as Chinchilla, on 1.4 trillion tokens for efficiency reasons
- Chinchilla is compared to Gopher and other large language models, showing that Chinchilla's smaller size leads to reduced memory footprint and inference cost while maintaining similar performance

4.1. Model and training details

- Chinchilla is trained with a specific set of hyperparameters (detailed in Table 4) and shares the same model architecture and training setup as Gopher with some exceptions
- Chinchilla is trained on the MassiveText dataset but with a slightly different subset distribution to accommodate the increased number of training tokens
- The AdamW optimizer is used for Chinchilla instead of Adam, resulting in improved language modeling loss and downstream task performance
- Chinchilla employs a modified SentencePiece tokenizer without NFKC normalization, which helps improve the representation of certain content like mathematics and chemistry
- While the forward and backward passes are computed in bfloat16, Chinchilla stores a float32 copy of weights in the distributed optimizer state
- All models, including Chinchilla, are trained on TPUv3/TPUv4 using JAX and Haiku.
- Chinchilla's evaluation includes a range of language modeling and downstream tasks (see Table 5) for comparison with previous work

4.2. Results

- **Performance on The Pile:** Chinchilla consistently outperforms Gopher across various evaluation sets in The Pile dataset, as indicated by bits-per-byte (bpb) improvements
- **Language Modeling Benchmarks:** Chinchilla achieves a lower perplexity (7.16) on Wikitext103 compared to Gopher (7.75), although it should be noted that Chinchilla is trained on significantly more data, raising concerns of train/test set leakage
- **Massive Multitask Language Understanding (MMLU):** Chinchilla significantly outperforms Gopher on the MMLU benchmark, with an average 5-shot accuracy of 67.6%, representing a 7.6% improvement over Gopher. Chinchilla also surpasses an expert forecast for June 2023 accuracy
- **Task-Specific Performance:** Chinchilla generally improves performance on a wide range of tasks compared to Gopher. However, it falls short on a few tasks such as college_mathematics, econometrics, moral_scenarios, and formal_logic
- **Word Prediction and Reading Comprehension:** Chinchilla achieves higher accuracy on the LAMBADA word prediction dataset (77.4%) compared to Gopher (74.5%). It also significantly improves performance on RACE-h and RACE-m reading comprehension tasks
- **BIG-bench Tasks:** Chinchilla outperforms Gopher on the majority of BIG-bench tasks, resulting in an average performance improvement of 10.7%
- **Common Sense Benchmarks:** Chinchilla excels on common sense benchmarks, outperforming both Gopher and GPT-3 on most tasks
- **Closed-Book Question Answering:** Chinchilla achieves new closed-book state-of-the-art (SOTA) accuracies on the Natural Questions dataset, surpassing Gopher's performance by a substantial margin. It also outperforms GPT-3 on TriviaQA

- **Risks and Ethical Concerns:** Chinchilla, like Gopher, carries risks such as biases, generation of potentially offensive content, and privacy concerns. These concerns remain challenging to address comprehensively
- **Limitations:** Chinchilla underperforms on certain tasks, and there are concerns about data leakage due to its larger training dataset. Additionally, the study emphasizes the importance of mitigating ethical risks associated with large language models

Discussion & Conclusion

- The trend in large language model training involves increasing model size without a proportional increase in training tokens
- This trend may lead to underperformance compared to what could be achieved with the same compute budget
- Three predictive approaches are proposed based on over 400 training runs, all indicating that Gopher is oversized, and smaller models trained on more data would perform better
- Chinchilla, a 70B parameter model, was created and outperformed Gopher and larger models in various evaluation tasks
- Limitations include the absence of intermediate-scale training runs, assumptions about the relationship between compute budget, model size, and training tokens, and training runs with less than one epoch of data
- There is a need for an increased focus on scaling datasets with a strong emphasis on quality
- Ethical and privacy concerns arise when training on trillions of tokens, necessitating dataset introspection and research into the interaction between model performance and toxicity
- The trade-off between model size and data amount is likely applicable to other modalities beyond language models
- The proposed methods for choosing optimal model size and training steps are considered applicable and reproducible in various settings

Article #9 Notes: Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster

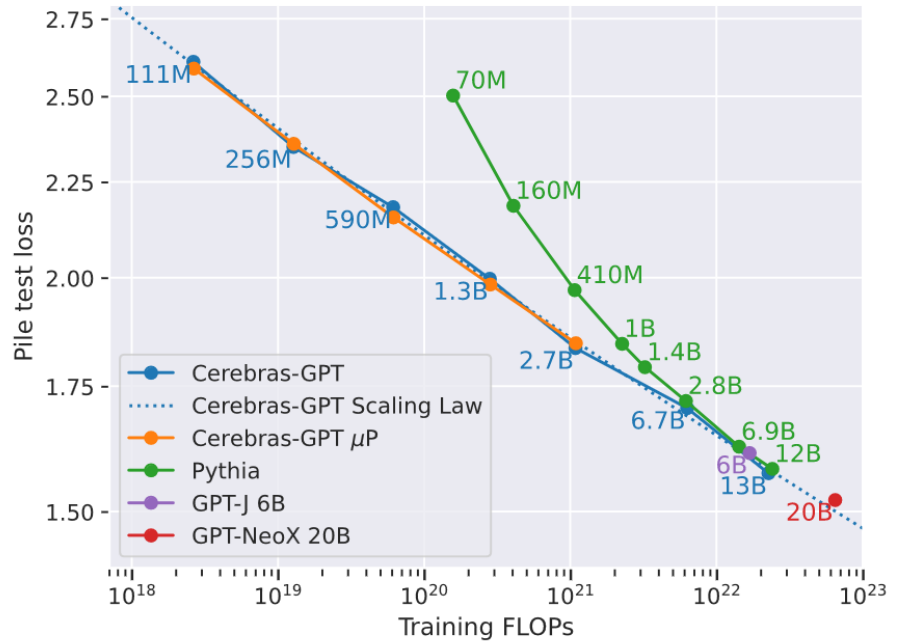
Source Title	Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster
Source citation (APA Format)	Dey, N., Gosal, G., Zhiming, Chen, Khachane, H., Marshall, W., Pathria, R., Tom, M., & Hestness, J. (2023). <i>Cerebras-GPT: Open compute-optimal language models trained on the Cerebras wafer-scale cluster</i> (arXiv:2304.03208). arXiv. http://arxiv.org/abs/2304.03208
Original URL	http://arxiv.org/abs/2304.03208
Source type	Journal Article
Keywords	Artificial Intelligence, Large Language Models (LLMs), Open Language Models, Generative AI, Compute-Optimal AI, Cerebras
#Tags	#ai, #llm, #opensource, #efficientscaling
Summary of key points + notes (include methodology)	The article presents research that advances large language models through efficient pre-training and scaling. It covers the training of Cerebras-GPT, a family of open source compute-optimal language models scaled from 111M to 13B parameters, trained on the Eleuther Pile dataset following scaling laws from DeepMind's Chinchilla. It also explores how Maximal Update Parameterization (μP) can improve model scaling as well as Andromeda, the specialized supercomputer used by Cerebras to train the model. The paper sheds light on the predictable power-law scaling of these models and provides detailed instructions for reproducing the results. The architecture of the Cerebras Wafer-Scale Cluster, designed for large-scale parallel deep learning training, is described, with a particular focus on the Cerebras Stack and the Weight Streaming mode, which eliminates the need for complex data and model parallelism, enhancing performance at small per-system batch sizes. After training the model, it showcased Cerebras-GPT's exceptional efficiency in both pre-training and downstream tasks in comparison to other models. Finally, the researchers also acknowledge the limitations of the model and identified research directions, such as investigating position embeddings like RoPE and ALiBi, activation functions such

as SwiGLU, and training paradigms like denoising pre-training objectives, instruction fine-tuning, and dataset cleaning.

Research Question/Problem/Need

How can scaling techniques be applied to train a compute-optimal large language model?

Important Figures

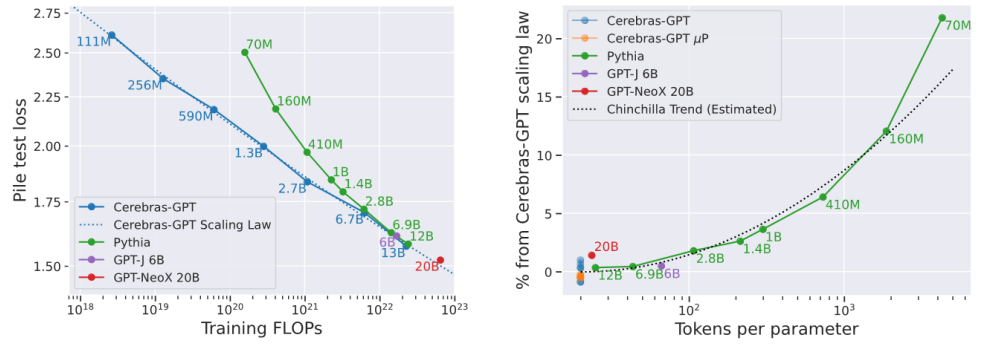


Pile test set loss given pre-training FLOPs for Cerebras-GPT, GPT-J, GPT-NeoX, and Pythia. Cerebras-GPT requires less training FLOPs (significantly less towards the left of the graph) to achieve similar losses to the other models).

Table 1: Cerebras-GPT model architecture and training algorithm details

Parameters	Model Dimensions				Total tokens	Batch Size (tokens)	Learning Rate (LR)	LR Decay Type
	d_{model}	n_{layers}	d_{head}	d_{ffn}				
111M	768	10	64	3072	2.2B	246K	6.0E-04	Linear
256M	1088	14	64	4352	5.1B	541K	6.0E-04	Linear
590M	1536	18	128	6144	11.8B	541K	2.0E-04	Linear
1.3B	2048	24	128	8192	26.3B	1.08M	2.0E-04	Cosine
2.7B	2560	32	80	10240	53.0B	1.08M	2.0E-04	Cosine
6.7B	4096	32	128	16384	133.2B	2.13M	1.2E-04	Linear
13B	5120	40	128	20480	257.1B	1.47M→2.21M	1.2E-04	Cosine
111M + μ P	768	10	64	3072	2.2B	246K	6.0E-03	Linear
256M + μ P	1088	14	64	4352	5.1B	541K	6.0E-03	Linear
590M + μ P	1536	18	128	6144	11.8B	541K	6.0E-03	Linear
1.3B + μ P	2048	24	128	8192	26.3B	1.08M	6.0E-03	Linear
2.7B + μ P	2560	32	80	10240	53.0B	1.08M	6.0E-03	Linear

Cerebras-GPT model architecture and training algorithm details



Again, as highlighted by the graph on the left, Cerebras-GPT requires significantly less training FLOPs to achieve similar loss on the Pile dataset as other models. Additionally, Figure 3 shows that for optimal pre-training on the Pile, using approximately 20 tokens per parameter is consistent with Chinchilla results on the MassiveText dataset, as it indicates a predictable percentage loss increase compared to the Cerebras-GPT frontier.

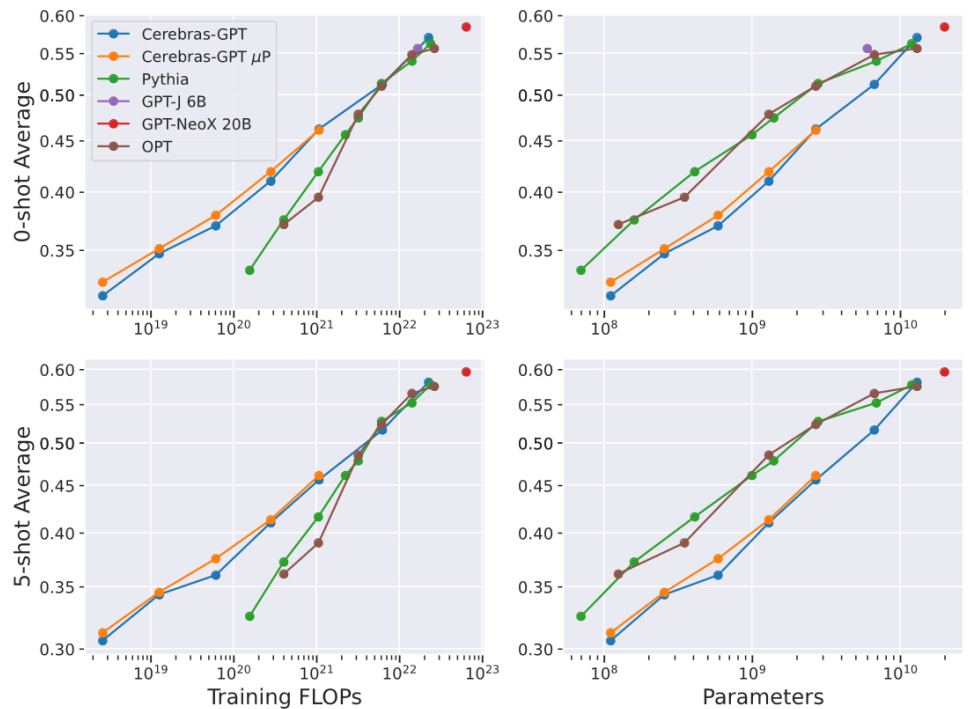


Figure 4 illustrates that Cerebras-GPT models establish the compute-optimal frontier for downstream tasks, with the 13B model exhibiting the best average results for models of comparable size, and it also demonstrates that downstream accuracy is predictable by model size for models trained with fixed tokens-per-parameter, implying competitiveness with GPT-NeoX 20B if scaled accordingly.

Table 2: Zero-shot downstream task results for large publicly-available models. Full results in Table 8

Model	Pre-training (↓)		Downstream task accuracy (↑)								
	Training FLOPs	Pile test xent	Hella-Swag	PIQA	Wino-Grande	Lambda	ARC-e	ARC-c	Open-BookQA	Downstream Avg.	
OPT	2.7B	6.1e21	-	0.458	0.738	0.610	0.637	0.609	0.268	0.250	0.510
Pythia	2.8B	6.1e21	1.720	0.451	0.737	0.612	0.654	0.629	0.288	0.220	0.513
Cerebras-GPT	2.7B	1.1e21	1.834	0.386	0.701	0.559	0.567	0.571	0.246	0.206	0.462
GPT-J	6.1B	1.7e22	1.613	0.518	0.752	0.640	0.683	0.670	0.340	0.288	0.556
OPT	6.7B	1.4e22	-	0.505	0.763	0.654	0.677	0.656	0.307	0.276	0.548
Pythia	6.9B	1.4e22	1.626	0.482	0.746	0.611	0.679	0.669	0.323	0.270	0.540
Cerebras-GPT	6.7B	6.3e21	1.704	0.447	0.739	0.602	0.636	0.643	0.282	0.238	0.512
OPT	13B	2.7e22	-	0.524	0.759	0.651	0.687	0.671	0.329	0.270	0.556
Pythia	12B	2.4e22	1.582	0.505	0.761	0.645	0.705	0.700	0.336	0.284	0.562
Cerebras-GPT	13B	2.3e22	1.572	0.513	0.766	0.646	0.696	0.714	0.367	0.286	0.570
GPT-NeoX	20B	6.4e22	1.519	0.535	0.779	0.661	0.720	0.723	0.380	0.290	0.584
Pythia	2.8B	6.1e21	1.724	0.466	0.743	0.612	0.672	0.662	0.299	0.232	0.526
Pile-dedup	6.9B	1.4e22	1.644	0.488	0.756	0.636	0.695	0.667	0.320	0.252	0.545
	12B	2.4e22	1.601	0.516	0.761	0.639	0.712	0.697	0.341	0.280	0.564

Table 2 presents detailed comparisons of large models, highlighting the best performers for various tasks and model sizes, including Pythia models trained on a deduplicated Pile. Despite challenges in pre-training, Pythia models generally show improved downstream task accuracy (1.8% on average), suggesting the potential benefits of deduplication.

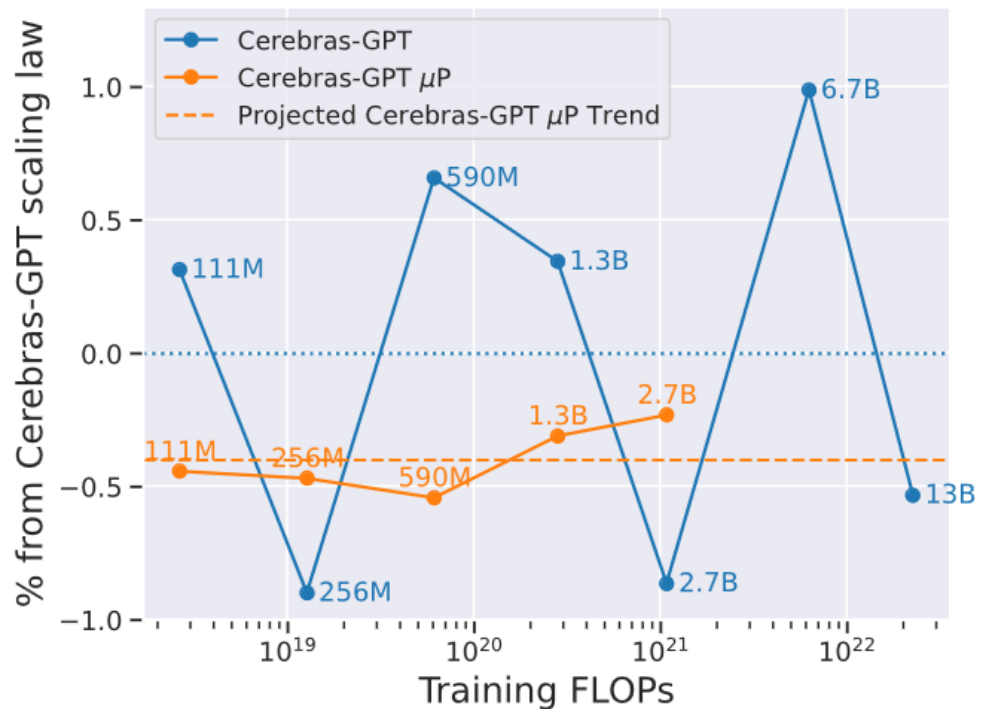


Figure 5 demonstrates that μP models exhibit more predictable scaling with an average improvement of 0.43% in Pile test loss compared to Cerebras-GPT SP models, along with substantially lower variance (0.04% vs. 0.66%), highlighting the model's robustness and reliability

Table 3: Pile pre-training test loss and zero-shot downstream task results for μ P and SP models.

Model	Pre-train Pile test xent (\downarrow)	Downstream task accuracy (f)								
		Hella-Swag	PIQA	Wino-Grande	Lambada	ARC-e	ARC-c	Open-BookQA	Down-stream Average	
Cerebras-GPT	111M	2.608	0.268	0.594	0.488	0.194	0.380	0.166	0.118	0.315
Cerebras-GPT + μ P	111M	2.588	0.268	0.598	0.519	0.204	0.390	0.176	0.124	0.325
Cerebras-GPT	256M	2.349	0.274	0.613	0.511	0.293	0.410	0.170	0.158	0.347
Cerebras-GPT + μ P	256M	2.359	0.274	0.617	0.505	0.287	0.427	0.194	0.156	0.351
Cerebras-GPT	590M	2.181	0.291	0.627	0.498	0.366	0.464	0.190	0.158	0.370
Cerebras-GPT + μ P	590M	2.155	0.295	0.644	0.517	0.362	0.470	0.194	0.172	0.379
Cerebras-GPT	1.3B	1.997	0.325	0.664	0.521	0.462	0.508	0.224	0.166	0.410
Cerebras-GPT + μ P	1.3B	1.984	0.334	0.682	0.512	0.471	0.515	0.223	0.196	0.419
Cerebras-GPT	2.7B	1.834	0.386	0.701	0.559	0.567	0.571	0.246	0.206	0.462
Cerebras-GPT + μ P	2.7B	1.846	0.388	0.697	0.557	0.558	0.569	0.241	0.218	0.461

Table 3 demonstrates that μ P models consistently enhance downstream performance, with an average 1.7% relative improvement in tasks compared to SP models, except for a specific 2.7B parameter model, where SP models unexpectedly performed well, while the 2.7B + μ P model remained competitive

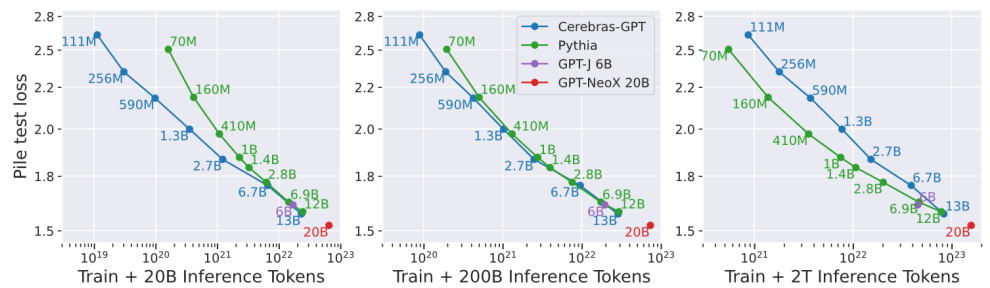
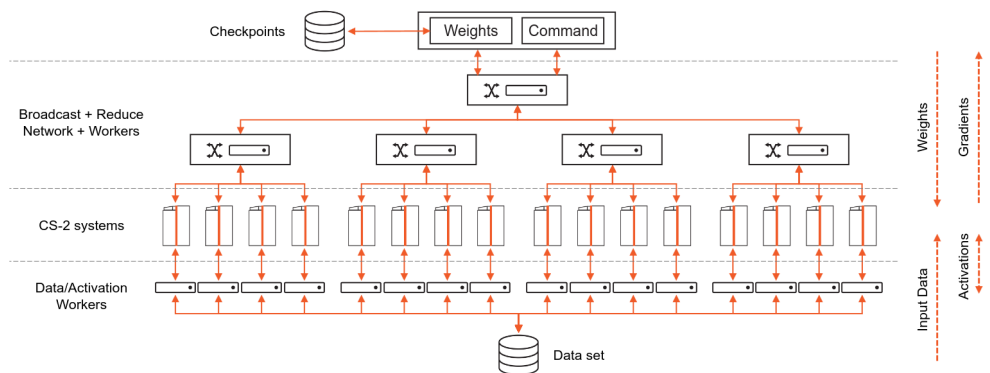


Figure 6 highlights that most Cerebras-GPT models outperform Pythia models in terms of Pile test loss per compute FLOP until a threshold of about 200B inference tokens, suggesting a trade-off where models trained with token counts between Cerebras-GPT and Pythia frontiers may offer better loss for the same compute budget



Logical architecture of the Cerebras Wafer-Scale Cluster

Table 4: Andromeda weak scaling tests show linear performance scaling up to 16 CS-2s

Model	Sequence Length	Per CS-2 Batch Size	Performance relative to 1 CS-2			
			2 CS-2s	4 CS-2s	8 CS-2s	16 CS-2s
GPT-3 XL 1.3B	2,048	121	1.99x	3.94x	7.87x	15.50x
GPT-3 XL 1.3B	10,000	33	1.99x	3.97x	7.95x	15.87x
GPT-3 2.7B	2,048	121	1.98x	3.91x	7.86x	15.62x
GPT-3 6.7B	2,048	85	1.99x	3.89x	7.91x	15.45x
GPT-3 20B	2,048	50	1.92x	3.75x	7.93x	15.32x
GPT-J 6B	2,048	65	1.97x	3.65x	7.69x	14.52x
GPT-NeoX 20B	2,048	50	1.98x	3.92x	8.05x	15.45x

Table 4 demonstrates that Andromeda achieves remarkable linear scaling, within 9%, for all model sizes and CS-2 system counts when applying a weak scaling approach, where batch size increases proportionally with the number of systems, as training steps progress, indicating impressive performance and scalability

Table 5: Strong scaling performance for batch sizes used to train larger models. To get to the user's full batch size, CSOft uses data parallelism across systems and gradient accumulation.

Model	Batch Size	Performance relative to 1 CS-2 (per CS-2 batch)			
		1 CS-2	2 CS-2s	4 CS-2s	8 CS-2s
1.3B	528	1.0x (132)	1.99x (132)	3.97x (132)	7.10x (66)
2.7B	528	1.0x (88)	1.99x (88)	3.77x (66)	7.43x (66)
6.7B	1,040	1.0x (65)	1.99x (65)	3.97x (65)	7.90x (65)
13B	1,040	1.0x (65)	1.99x (65)	3.95x (65)	7.84x (65)

Table 5 highlights that Andromeda maintains high utilization even during strong scaling of batch sizes, where fixed batch sizes are distributed across different numbers of Andromeda systems. The results indicate consistent performance scalability for batch sizes commonly used with these models, demonstrating the system's efficient utilization and adaptability to varying batch sizes

Table 6: Andromeda FLOP/s utilization relative to 1 CS-2 training the 111M parameter model. Here, larger values mean higher utilization.

Model	Batch Size	Number of CS-2s	Per CS-2 Batch Size	Relative Utilization
111M	120	1	120	1.00
256M	264	1	264	1.00
590M	264	1	264	0.92
1.3B	528	4	132	0.96
2.7B	528	4	132	0.96
6.7B	1040	16	65	1.05
13B	1080	12	45	1.02

Table 6 shows consistent FLOP/s utilization across various model sizes, with performance deviating by less than 8%, highlighting Andromeda's robust scalability and consistent performance across different setups

VOCAB: (w/definition)

Autoregressive Transformer Decoder Model: A neural architecture commonly used for sequential data generation tasks, such as language generation and machine translation, where it predicts tokens one at a time based on previously generated tokens, making it well-suited for tasks involving ordered data.

Deduplication: A method of elimination duplicate copies of repeating data

	<p>Standard Parameterization: A commonly used initialization approach where model weights are initialized with specific standard deviations, promoting stability and predictable behavior during training and inference. It serves as a foundational configuration for large models before further optimization or fine-tuning.</p> <p>Maximal Update Parameterization: An initialization and hyperparameter tuning approach for large language models that enhances training stability, control over initialization, layer-wise learning rates, and activation magnitudes, ultimately improving the transferability of training hyperparameters from smaller to larger scale models. It addresses challenges faced by Standard Parameterization (SP) when scaling large models, resulting in more stable and predictable training behavior.</p> <p>Pre-training: The initial phase of model training in which a neural network is exposed to a massive amount of text data to learn general language understanding and context. During this phase, the model learns to predict the next word or token in a sentence, capturing linguistic patterns, semantics, and world knowledge, which it can later apply to specific natural language processing tasks through fine-tuning.</p> <p>Scaling: The process of increasing the size, capacity, or complexity of a neural network, typically by adding more data, parameters, layers, or computational resources</p> <p>Downstream Results: The model's performance and effectiveness when applied to specific natural language processing tasks, such as text classification, language translation, sentiment analysis, or question answering, after the initial pre-training phase. These results assess how well the model can leverage its learned language understanding to solve real-world language-related challenges.</p>
<p>Cited references to follow up on</p>	<p>Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling Language Modeling with Pathways, 2022. URL https://arxiv.org/abs/2204.02311</p> <p>Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling Vision Transformers to 22 Billion Parameters, 2023. URL https://arxiv.org/abs/2302.05442</p> <p>Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile: An 800GB Dataset of Diverse Text for Language Modeling, 2020. URL https://arxiv.org/abs/2101.00027</p> <p>Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In <i>Advances in Neural Information Processing Systems</i>, 2017.</p>
<p>Follow up Questions</p>	<p>Is the Andromeda supercomputer and the Wafer-Scale cluster more energy efficient than other training hardware?</p> <p>What were the carbon emissions throughout the training lifecycle of</p>

Cerebras-GPT?

How was the Pile Dataset scaled?

Are there environmental benefits to Maximal Update Parameterization (μP)?

How specifically can organizations and government better analyze training costs and carbon emissions as mentioned in the paper?

Introduction

- Recent research shows advances that can vastly improve LLM quality and efficiency
 - Scaling laws
 - Training on more data
 - Maximal Update Parameterization
- The research community has trained and released many open-source models with state-of-the-art efficiency for their size, but they aren't compute-efficient
- Cerebras-GPT is an effort to combine efficient scaling techniques to produce compute-optimal pre-trained models and the corresponding scaling laws
- Overall, the contributions of this work includes the following
 - Trained Cerebras-GPT compute-optimal models scaled from 111M to 13B parameters to collect compute-efficient scaling laws
 - Showed that these models provide state-of-the-art pre-training efficiency on both pre-training and downstream objectives compared to other open models
 - Provided detailed instructions how to reproduce results, including the use of μP
 - Documented the experience training on the Andromeda AI Cluster, comprising 16 Cerebras CS-2 systems

Methodology

Model Architecture

- Cerebras-GPT models have a GPT-3-like architecture, an autoregressive transformer decoder model
- Unlike GPT-3, Cerebras-GPT uses dense attention in all decoder blocks

Pre-training Corpus

- Models are pre-trained on the Pile dataset, which consist of data from 22 sources
- The corpora is tokenized with byte-pair encoding and the GPT-2 vocabulary
- Deduplication wasn't performed but it could further improve results

Model Training

- Model configurations
 - AdamW optimizer with $(\text{beta1}, \text{beta2}) = (0.9, 0.95)$
 - Epsilon $1\text{e-}8$ for small models and $1\text{e-}9$ for 6.7B and 13B parameter models
 - Weight decay of 0.1
 - Gradient norm clipping of 1.0
 - Learning rates and batch sizes consistent with prior works
 - Linear learning rate over cosine decay more often because it tends to perform better
- Cerebras-GPT was scaled following the DeepMind Chinchilla scaling methodology
- This paper is the first to estimate the compute-efficient tokens per parameter for the Pile dataset
- The models were trained with both FP16 mixed precision and bfloat16 precision
 - Bfloat16 more stable generally due to extra exponent range

Standard (SP) and Maximal Update Parameterization (μ P)

- The main Cerebras-GPT models are configured with the common standard parameterization (SP) approach
 - Weights initialized from normal distributions with constant standard deviation or standard deviation based on the shape of each layer
 - Embedding and hidden layer weights are initialized with a truncated normal distribution with standard deviation $\sigma = 0.02$
 - Standard deviation of $\sigma = 0.02/\sqrt{2 \cdot n_{layers}}$ for the last layer, following the GPT-2 initialization
- SP models tend to become unstable as they scale

Maximal Update Parameterization

- They trained a set of Cerebras-GPT models with Maximal Update Parameterization (μ P) to address the issues of SP
- μ P increases training stability by controlling initialization, layer-wise learning rates, and activation magnitudes and improves the transferability of training hyperparameters from smaller to larger scale models (μ Transfer)
- A smaller set of Cerebras-GPT models was trained using μ P, and the tuned hyperparameters were transferred along the μ P scaling law to a 2.7B parameter model

Results

- Cerebras-GPT models are shown to define the state-of-the-art compute-optimal Pareto frontier on both pre-training and downstream objectives

Pre-training Results

- They scaled and pre-trained Cerebras-GPT models from 111M-13B parameters on the Pile dataset
- Refer to Figure 2 and Figure 3 annotations

Downstream Results

- Cerebras-GPT and other publicly-available models were evaluated on a suite of seven common sense reasoning tasks using the EleutherAI evaluation harness
- Cerebras-GPT models form the compute-optimal Pareto frontier for downstream tasks as well
- Refer to Figure 4 annotations

Maximal Update Parameterization (μ P) and μ Transfer

- Scaling Cerebras-GPT models with SP resulted in challenges predicting appropriate hyperparameters and substantial variance around their common scaling law
- μ P models had an average of 0.43% improved Pile test loss and 1.7% higher average downstream task accuracy compared to SP models; μ P performance also scaled more predictably
- Refer to Figure 5 and Table 3 annotations

Trading Off Training and Inference FLOPs

- The analysis has primarily focused on compute-optimal pre-training, where compute cost is tied to model size squared, but there is growing interest in considering model inference costs, indicating that smaller models trained on more tokens can offer substantial loss improvements and inference cost advantages proportional to their size
- They propose a technique to identify training and inference compute-optimal frontiers that practitioners could use to estimate how models should be pre-trained considering deployment costs

$$F = f_{\text{pre-train_total}} + n_{\text{infer_tokens}} \cdot f_{\text{infer/token}}$$

$$\propto \mathcal{O}(p^2) + n_{\text{infer_tokens}} \cdot \mathcal{O}(p)$$

F is total compute cost, *f* represents FLOPs costs for full pre-training and per-token infer, $n_{\text{infer_tokens}}$ is the number of expected inference tokens, and *p* is the parameter count

- Organizations and governments can better assess the total costs when budgeting large-scale training runs
- Simple analysis can be applied to monetary, energy, or carbon footprint costs as well

Cerebras Stack

- All studies were run on the Cerebras Wafer-Scale Cluster named “Andromeda”, which contains 16 Cerebras CS-2 systems

Andromeda AI Supercomputer

- Andromeda is a Cerebras Wafer-Scale Cluster with 16 CS-2 systems, each containing a WSE-2 processor with 40 GB of SRAM and 7.5 PetaFLOP/s peak throughput
- Andromeda’s architecture is designed for large-scale parallel deep learning training, with a total peak throughput of 120 PFLOP/s
- Weights and command servers manage computation by broadcasting weights and control instructions and collecting and reducing gradients
- Activation workers handle input data and activations, reading data from disk, creating subbatches for training, and managing activation checkpointing when required

CSoft Platform and Weight Streaming Mode

- Andromeda uses the CSoft to run deep learning applications, with models written and trained in both TensorFlow and PyTorch
- CSoft handles model compilation, orchestration, and optimizations like subbatch sizing, gradient accumulation, activation recomputation, and data layouts for high performance
- The Weight Streaming mode, shown in Figure 7, enables training models limited only by the memory capacity of weight servers, allowing testing beyond the GPT-3 175B parameter model without additional changes

- Unlike existing accelerator execution modes, Weight Streaming eliminates the need for complex data and model parallelism, offering solid performance at small per-system batch sizes by moving weights to the wafer and gradients from the wafer

Performance Scalability

- Andromeda provides near-linear performance scaling up to the full 16 CS-2s
- Refer to annotations

Limitations

- This work focuses on training foundational models without exploring recent architectural features, downstream task tuning procedures, or dataset cleaning methods used in contemporary works
- Future research directions include investigating position embeddings like RoPE and ALiBi, activation functions such as SwiGLU, and training paradigms like denoising pre-training objectives and instruction fine-tuning
- Dataset cleaning is identified as a potential area for improvement, as demonstrated by Pythia models showing enhanced downstream task accuracy when trained on deduplicated data
- Cerebras-GPT models have not undergone extensive testing in downstream tasks or real-world applications, and further safety-related testing, mitigations, and output curation are necessary before deployment

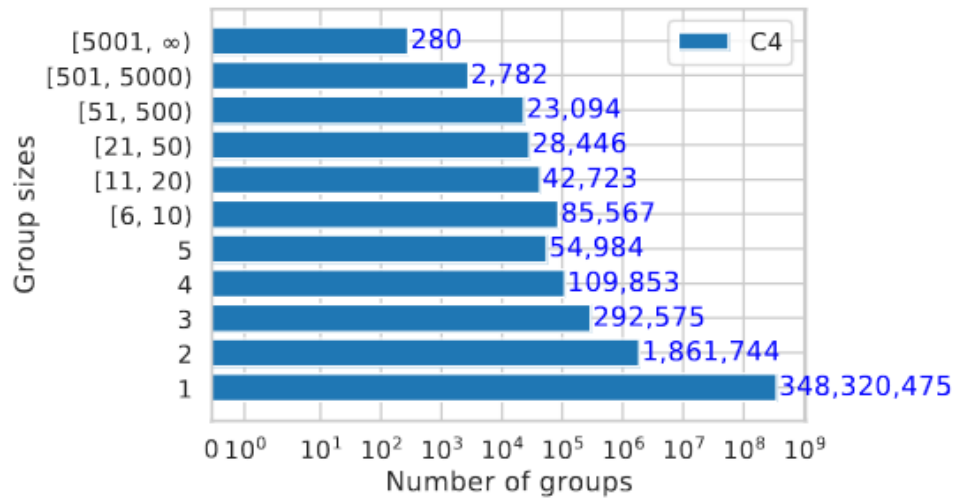
Article #10 Notes: Deduplicating Training Data Makes Language Models Better

Source Title	Deduplicating Training Data Makes Language Models Better
Source citation (APA Format)	Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., & Carlini, N. (2021). <i>Deduplicating training data makes language models better</i> (arXiv:2107.06499). arXiv. http://arxiv.org/abs/2107.06499
Original URL	https://arxiv.org/abs/2107.06499
Source type	Journal Article
Keywords	Artificial Intelligence, Large Language Models (LLMs), Datasets, Big Data, Deduplication
#Tags	#ai, #llm, #nlp, #nlg, #datasets, #bigdata
Summary of key points + notes (include methodology)	<p>Recently, large language models have been growing larger and larger in size. As such, the corporas of text needed to train these models has grown significantly as well. However, large datasets, especially those that span numerous terabytes in size, are hard to create manually and quality test to ensure no biases and high-quality data. This paper explored the specific effect of deduplication, or removing repetitive and redundant training examples from a dataset. The researchers observed various benefits, from reducing memorized data to more efficient model training and sizes, and no downsides to deduplication. The paper proceeds to outline how data was deduplicated with two primary methods and examined the extent of duplicate content in the dataset and the effect of deduplicated data on models. The first method of deduplicated detailed was using a suffix array to remove duplicate substrings if they appear verbatim in more than one example, and the second method detailed using n-gram similarity between all pairs of examples and removing those that had high overlap. After carrying out deduplication, various resultant metrics were provided.</p>
Research Question/Problem/Need	How does deduplicating training data affect language models?

Important Figures

Dataset	Example	Near-Duplicate Example
Wiki-40B	\n_START_ARTICLE_\nHum Award for Most Impactful Character \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]	\n_START_ARTICLE_\nHum Award for Best Actor in a Negative Role \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]
LM1B	I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .	I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .
C4	Affordable and convenient holiday flights take off from your departure country, "Canada". From May 2019 to October 2019, Condor flights to your dream destination will be roughly 6 a week! Book your Halifax (YHZ) - Basel (BSL) flight now, and look forward to your "Switzerland" destination!	Affordable and convenient holiday flights take off from your departure country, "USA". From April 2019 to October 2019, Condor flights to your dream destination will be roughly 7 a week! Book your Maui Kahului (OGG) - Dubrovnik (DBV) flight now, and look forward to your "Croatia" destination!

Examples of near-duplicates identified by NearDup, the approximate matching algorithm, from each dataset



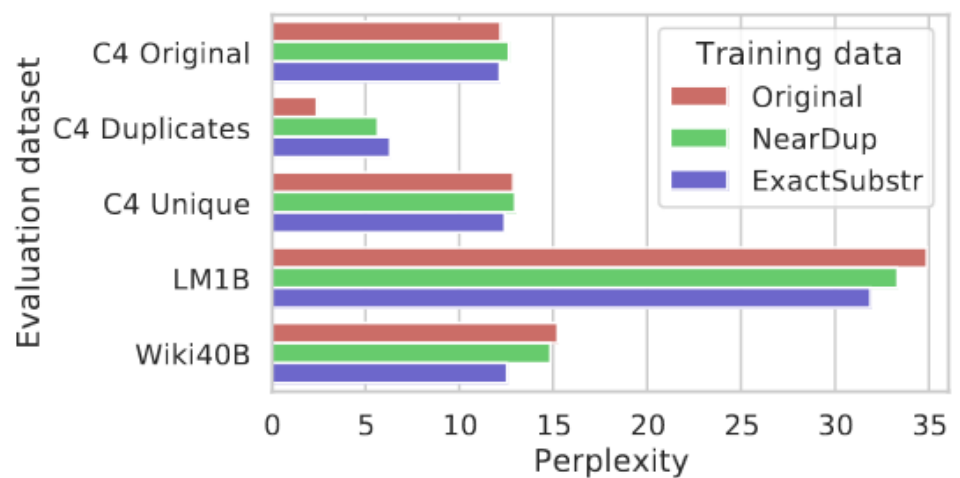
The distribution of near-duplicate cluster sizes from running NearDup on C4

	% train examples with dup in train	% valid with dup in valid	% valid with dup in train
C4	3.04%	1.59%	4.60%
RealNews	13.63%	1.25%	14.35%
LM1B	4.86%	0.07%	4.92%
Wiki40B	0.39%	0.26%	0.72%

The fraction of examples identified by NearDup as duplicates

	% train tokens with dup in train		% valid with dup in train
C4	7.18%	0.75 %	1.38 %
RealNews	19.4 %	2.61 %	3.37 %
LM1B	0.76%	0.016%	0.019%
Wiki40B	2.76%	0.52 %	0.67 %

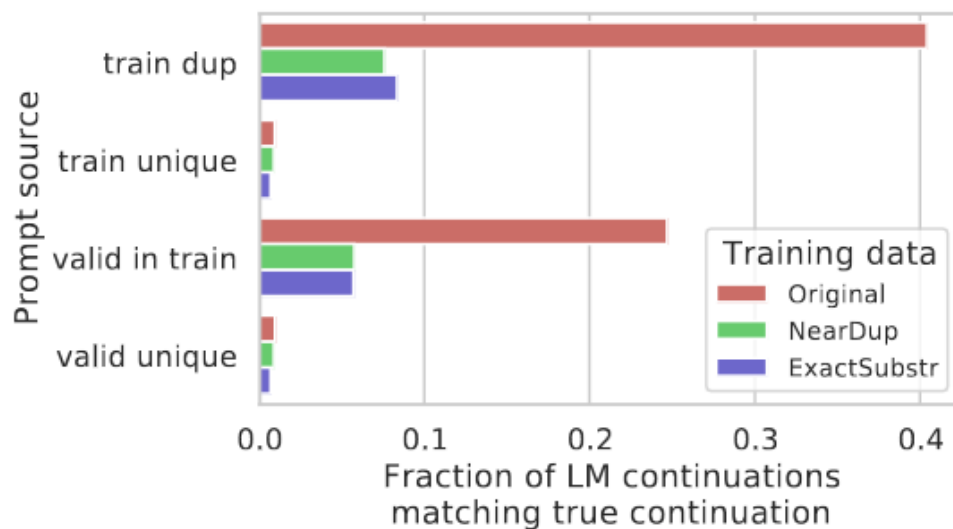
The fraction of examples identified by ExactSubstr as part of an exact duplicate 50-token substring



Impact of deduplicating the training set on validation perplexity of trained models

Model	1 Epoch	2 Epochs
XL-ORIGINAL	1.926%	1.571%
XL-NEARDUP	0.189%	0.264%
XL-EXACTSUBSTR	0.138%	0.168%

When generating 100k sequences with no prompting, over 1% of the tokens emitted from a model trained on the original dataset are part of a 50-token long sequence copied directly from the training dataset. This drops to 0.1% for the deduplicated datasets.



The proportion of generations which have edit similarity above 0.8 with the groundtruth continuation when using the LM to generate continuations for 32-token prompts identified by NEARDUP as either duplicated or unique.

Model	Dataset	Orig	Dups	Unique
Transformer-XL	LM1B	21.77	10.11	23.58
GROVER-Base	RealNews	15.44	13.77	15.73
GROVER-XL	RealNews	9.15	7.68	9.45

For each model, the perplexity of the official validation set (Orig), valid set examples which were identified by NEARDUP as matches of train set examples (Dups), and valid set examples identified by NEARDUP as unique (Unique).

VOCAB: (w/definition)

Exact substring matching: Exact substring matching is a process of identifying and eliminating duplicated substrings within a dataset by comparing sequences to find shared substrings of a minimum specified length, such as 50 BPE tokens in this case, and then removing one occurrence of the shared substring. This technique is used to reduce redundancy in text data.

Approximate full document matching: Approximate full document matching, known as NEARDUP, involves using the MinHash algorithm to identify documents that are nearly identical by comparing their n-gram sets and approximating the Jaccard Index. If the Jaccard Index is high enough, the documents are considered potential matches, and further similarity metrics, like edit similarity, can be employed to filter and identify duplicates, making it suitable for handling documents with slight variations.

Suffix array: A suffix array is a lexicographically-ordered list of all suffixes contained in a given sequence of text, facilitating efficient computation of substring queries and the identification of duplicated examples in linear time

	<p>MinHash: MinHash is an approximate matching algorithm that represents documents using sets of n-grams and utilizes hash functions to estimate the Jaccard Index, enabling the identification of potential matches between documents. It creates document signatures by sorting n-grams with a hash function, keeping the smallest hashed n-grams, and is commonly used in large-scale deduplication tasks, including the identification of duplicate documents.</p> <p>N-gram: An n-gram is a collection of n successive items in a text document that may include words, numbers, symbols, and punctuation</p> <p>N-gram similarity: N-gram similarity refers to the measurement of the similarity between two documents by comparing their sets of n-grams (contiguous sequences of n items, often words or characters) to estimate the Jaccard Index, which quantifies the overlap of n-grams between the documents, indicating their approximate similarity</p> <p>Jaccard Index: The Jaccard Index is a similarity measure used to assess the overlap or similarity between two sets by dividing the size of their intersection by the size of their union, producing a value between 0 and 1, where 1 indicates complete overlap or similarity and 0 indicates no common elements</p> <p>Train-test leakage: Train-test leakage, also known as data leakage, occurs when information from the test dataset is inadvertently used during the training of a machine learning model. This can lead to artificially inflated performance metrics during training but results in poor real-world performance since the model has gained knowledge it shouldn't have.</p> <p>Model perplexity: Model perplexity is a measurement of how well the model predicts a sequence of words. A lower perplexity score indicates that the model is better at predicting the next word in a sequence, suggesting a better understanding of the language.</p>
<p>Cited references to follow up on</p>	<p>Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. 2017. A closer look at memorization in deep networks. In <i>International Conference on Machine Learning</i>, pages 233–242. PMLR.</p> <p>Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. <i>Transactions of the Association for Computational Linguistics</i>, 6:587–604.</p>

	<p>Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? <i>arXiv preprint</i>.</p>
Follow up Questions	<p>What is the specific impact of deduplicated datasets on training compute efficiency?</p> <p>What are some other methods of ensuring data quality?</p> <p>What privacy concerns are associated with data memorization?</p> <p>How do train-test overlap issues, as discussed in the article, affect the performance and model selection in LLMs?</p> <p>How do different language model architectures and dataset sources influence the impact of deduplication on training efficiency and model performance?</p> <p>Can deduplication methods be combined?</p>

Notes (written with assistance from ChatGPT)

Introduction

- Recent progress in natural language processing is driven by large-scale text corpora used to train language models
- These datasets have grown significantly in size over the past few years, from gigabytes to terabytes
- Large datasets are challenging to curate manually, leading to potential quality issues and biases in trained models
- Duplicated training examples are a common source of bias in NLP datasets.
- The paper proposes two techniques for detecting and removing duplicated training data: exact substring matching and approximate full document matching
- Thorough deduplication of training data offers several advantages:
 - Reduces the rate of emitting memorized training data by a factor of 10×
 - Addresses train-test overlap issues, preventing overestimation of model accuracy and biasing model selection
 - Makes training more efficient and cost-effective, with datasets being up to 19% smaller
 - Does not negatively impact perplexity and can even reduce it by up to 10%.
- Data deduplication has no observed disadvantages
- The paper outlines the framework for text deduplication and examines the extent of duplicate content in common NLP datasets
- It also explores the impact of deduplication on test perplexity, the frequency of emitting memorized content, and the skewing of perplexity in existing models due to train-test overlap

Related Work

- The section discusses the use of large language model datasets and focuses on Transformer-based decoder-only language models used for open-ended text generation
- These models are typically trained on internet text, with examples like GPT-2 being trained on WebText and CommonCrawl
- Various models are mentioned, including GPT-3, GROVER, and T5, each trained on different variations of web data
- Some models are trained on curated internet sources, like Guo et al.'s model trained on processed Wikipedia text
- Non-English models use different datasets, like PANGU- α , which is trained on a non-public corpus of Chinese-language documents
- The section also highlights that some datasets are not publicly available
- It mentions deduplicating publicly available datasets, such as Wiki-40B, C4, RealNews, and the One Billion Word Language Model Benchmark
- The section raises concerns about contamination of downstream tasks when models are trained on internet datasets that overlap with evaluation datasets

- It discusses approaches to address contamination, such as removing overlapping documents from training sets
- The focus of the research mentioned is on the impact of duplicate text in language model training and validation sets on model perplexity and the presence of memorized content
- The section mentions the privacy risks associated with data memorization and highlights that some models emit over 1% of memorized training data
- Finally, it briefly mentions previous studies on duplicate text in other domains, such as code datasets and their impact on code understanding tasks

Language Modeling Datasets

- The section discusses the analysis of duplicate text in four datasets commonly used for training natural language generation systems, creating pre-trained models, and benchmarking language models
- The focus is on English datasets, but it's mentioned that similar issues could exist in non-English datasets
- The datasets analyzed include:
 - Wikipedia (Wiki-40B): This dataset consists of multi-lingual cleaned Wikipedia text, but the analysis focuses on the English portion. It contains 2.9 million Wikipedia pages with an average length of 768 BPE tokens.
 - One-Billion Word benchmark (LM1B): LM1B contains 30 million sentences of news commentary, with an average example length of 32 BPE tokens.
 - Colossal Cleaned Common Crawl (C4): C4 comprises 360 million web documents, with an average length of 486 BPE tokens. It was pre-processed to remove duplicates through a sophisticated deduplication process.
 - RealNews: RealNews is a subset of the Common Crawl containing articles from news domains. It includes 31 million documents with an average length of 793 BPE tokens. Deduplication in RealNews involved hashing the first 100 characters of each document and excluding documents with hash collisions, similar to C4.
- The section highlights the deduplication methods used for each dataset and the removal of duplicates based on various criteria like hash collisions and duplicate URLs

Methods for Identifying Duplicates

- Two complementary methods for deduplicating text data are introduced:
 - Exact Substring Duplication:
 - This method focuses on removing duplicate substrings from the dataset if they appear verbatim in more than one example
 - A minimum matching substring length of 50 tokens is chosen based on statistical analysis

- To efficiently identify duplicated training examples, a Suffix Array is constructed from the dataset, allowing for linear time substring queries
 - Approximate Matching with MinHash:
 - This method aims to remove entire duplicate examples from the dataset based on approximate matching
 - MinHash, an algorithm for estimating n-gram similarity, is used. Each document is represented by its set of n-grams, and hash functions are applied to approximate the Jaccard Index
 - A signature of size 9,000 and 5-grams are used in the implementation
 - Documents are considered potential matches if their Jaccard Index is sufficiently high, and further filtering is applied based on edit similarity
 - Clusters of similar documents are identified using a graph-based approach
- The section explains the principles and computational details of both deduplication methods

Deduplication Results

- The section presents the results of deduplication efforts on four datasets using two techniques
- Duplicate text across data splits was prioritized to be kept in the test or validation set and removed from the train set
- Results of deduplication:
 - Using NEARDUP, web-scrape datasets contain between 3.04% (C4) and 13.63% (RealNews) near duplicates. Wiki-40B has only 0.39% near duplicates in the train set
 - In C4, most near-duplicate clusters consist of single pairs, but some clusters have over 5,000 examples, with one cluster containing 250,933 examples
 - On average, EXACTSUBSTR removes more total content compared to NEARDUP, except for LM1B, where EXACTSUBSTR removes 8× less data due to shorter document lengths
 - NEARDUP and EXACTSUBSTR remove similar content; 77% of training examples removed by NEARDUP in C4 have at least one verbatim length-50 match found by EXACTSUBSTR
- Properties of duplicated text:
 - In C4 and Wiki-40B, much of the near-duplicated text appears to be computer-generated, with differences mainly in names, places, businesses, products, dates, etc
 - RealNews and LM1B, derived from news sites, have near-duplicates due to the same news articles appearing on multiple sites with slight formatting differences
- Train/Test Set Leakage:
 - Both deduplication methods identify overlap between train and validation sets
 - For example, 4.6% of the C4 validation set and 14.4% of the RealNews validation set have approximate duplicates in their respective training sets
 - This duplication poses problems for evaluation metrics, potentially inflating scores for models better at memorizing their training data
 - The effect of this leakage on publicly released models is evaluated in Section 6.3

Impact on Trained Models

- 1.5B parameter "XL" models, similar to GPT-2, were trained on C4-ORIGINAL, C4-NEARDUP, and C4-EXACTSUBSTR
- Three different random seeds of the 110M parameter "base" model were also trained on each of the three datasets, resulting in nine base-sized models
- All models used a Byte Pair Encoding (BPE) vocabulary trained on C4-NEARDUP with a 50K token budget
- Maximum sequence length during training was 512 tokens
- Model Perplexity:
 - Models' perplexity was computed on validation sets of LM1B, Wiki-40B, and subsets of the C4 validation set
 - Models trained on deduplicated data had significantly higher perplexity on validation set examples with duplicates in the training set compared to models trained on the original C4
 - EXACTSUBSTR-deduplicated models showed higher perplexity than NEARDUP-deduplicated models
 - Similar trends were observed for XL-sized models
- Generated Text:
 - Memorization tendencies were evaluated in text generation experiments with and without prompts
 - Without prompts, over 1% of generated tokens from XL-ORIGINAL belonged to memorized sub-sequences, while XL-EXACTSUBSTR and XL-NEARDUP had significantly less memorization
 - When prompts were used, models still tended to copy ground-truth text more often when the prompt came from a duplicate example, even for deduplicated models
- Impact on Existing Models:
 - Train-test leakage affected existing models, including Transformer-XL trained on LM1B and GROVER trained on RealNews
 - The presence of near-duplicates of the evaluation set in the train set significantly impacted model perplexity, with perplexity halving for some examples
 - Existing models also suffered from the issue of generating text from their train sets, with a significant portion of tokens in their outputs being part of verbatim matches in their training data

Discussion

- This paper focuses on analyzing data duplication in language model training datasets
- It quantifies the extent of data duplication, explores the impact of deduplication on model perplexity, and examines the reduction of memorized content

- Privacy implications of memorized training data are noted, emphasizing the need for careful consideration in dataset creation
- Different types of memorized text and potential negative consequences of deduplication are discussed
- The role of memorization in language models varies based on data nature and application, encouraging researchers to consider these factors
- The paper suggests future research directions, including methods to memorize or forget specific sequences based on application requirements

Conclusion

- Future language model research is encouraged to perform dataset deduplication
- Researchers can use the deduplicated datasets and tools provided in this study or develop new deduplication methods
- The specific deduplication technique used matters less than the act of performing stringent deduplication
- Deduplication generally does not harm model perplexity and can even improve it, despite smaller training datasets
- Avoiding duplicates between training and testing sets is crucial to prevent models from memorizing training data
- Deduplication contributes to reducing privacy concerns related to language models memorizing their training data

Article #11 Notes: Beyond Scale: the Diversity Coefficient as a Data Quality Metric Demonstrates LLMs are Pre-trained on Formally Diverse Data

Source Title	Beyond Scale: the Diversity Coefficient as a Data Quality Metric Demonstrates LLMs are Pre-trained on Formally Diverse Data
Source citation (APA Format)	Lee, A., Miranda, B., Sundar, S., & Koyejo, S. (2023). <i>Beyond scale: The diversity coefficient as a data quality metric demonstrates LLMs are pre-trained on formally diverse data</i> (arXiv:2306.13840). arXiv. http://arxiv.org/abs/2306.13840
Original URL	http://arxiv.org/abs/2306.13840
Source type	Journal Article
Keywords	Diversity Coefficient, Data Quality Metric, Large Language Models, Pre-training Data, Publicly Available Datasets
#Tags	#llm, #nlp, #datasets, #bigdata, #dataquality
Summary of key points + notes (include methodology)	This article discusses the quality of pre-training data for Large Language Models (LLMs) and how it is an important factor for training powerful LLMs. The authors propose to ground the discussion of data quality through the diversity coefficient, a data coverage metric that moves beyond scale alone. They extend the diversity coefficient to formally quantify data diversity of publicly available datasets and discover that LLMs are pre-trained on formally diverse data. The authors conclude that the diversity coefficient is reliable and can be used to build useful diverse datasets for LLMs. The methodology involves measuring the diversity coefficient of publicly available pre-training datasets to demonstrate that their formal diversity is high when compared to theoretical lower and upper bounds. The authors also conduct interpretability experiments to build confidence in the diversity coefficient and find that the coefficient aligns with intuitive properties of diversity, e.g., it increases as the number of latent concepts increases
Research Question/Problem/Need	How can the quality of pre-training data for LLMs be measured?

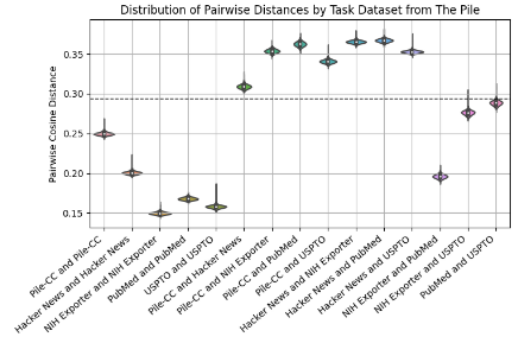
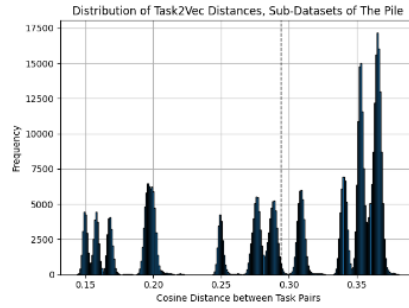
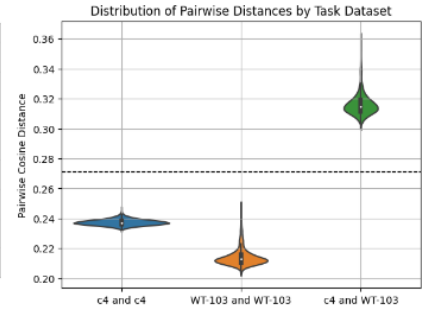
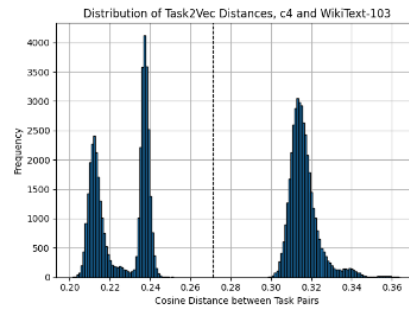
Important Figures

DATASET	DIVERSITY COEFF.
LOWER BOUND (LB)	0.0525 \pm 3.41E-4
NIH EXPORTER	0.15 \pm 3.218E-5
USPTO	0.1582 \pm 4.09E-5
PUBMED ABSTRACTS	0.168 \pm 2.63E-5
HACKERNEWS	0.201 \pm 4.52E-5
OPENWEBTEXT	0.222 \pm 1.00E-3
WIKITEXT-103	0.2140 \pm 7.93E-5
C4	0.2374 \pm 2.785E-5
SLIMPAJAMA	0.221 \pm 9.97E-4
OPENWEBTEXT	0.222 \pm 1.00E-3
THE PILE	0.2463 \pm 3.034E-5
PILE-CC	0.2497 \pm 3.41E-5
C4 AND WIKITEXT-103 (MIX1)	0.235 \pm 1.04E-3
UNION OF FIVE DATASETS (MIX2)	0.217 \pm 9.81E-4
UPPER BOUND (UB)	0.4037 \pm 1.932E-5

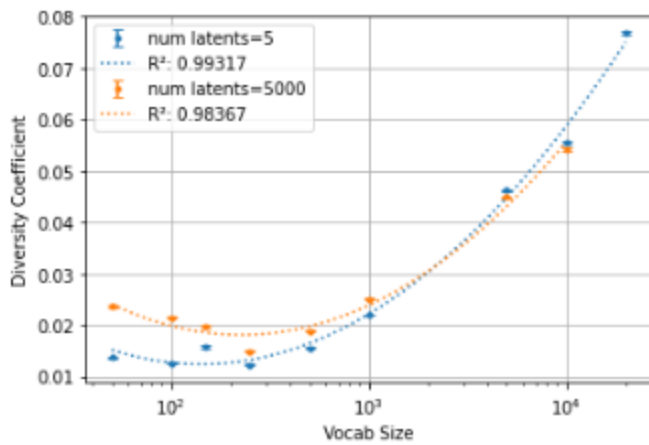
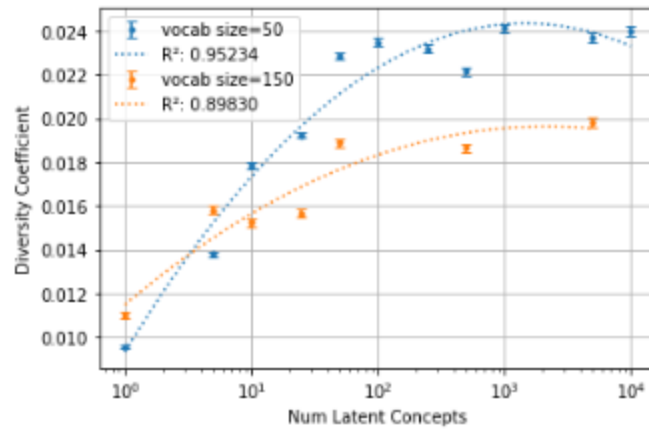
Diversity coefficients of LLM pre-training datasets with 95% confidence intervals are 2.7-4.76 times higher than the conceptual lower bound and more than half that of the upper bound.

DATASET	DIVERSITY COEFF.
LOWER BOUND (LB)	0.0525 \pm 3.41E-4
NIH EXPORTER	0.15 \pm 3.218E-5
USPTO	0.1582 \pm 4.09E-5
PUBMED ABSTRACTS	0.168 \pm 2.63E-5
HACKERNEWS	0.201 \pm 4.52E-5
WIKITEXT-103	0.2140 \pm 7.93E-5
C4	0.2374 \pm 2.785E-5
THE PILE	0.2463 \pm 3.034E-5
PILE-CC	0.2497 \pm 3.41E-5
C4 AND WIKITEXT-103	0.2711 \pm 3.22E-4
CONCATENATION OF FIVE DATASETS	0.2939 \pm 2.03E-4
UPPER BOUND (UB)	0.4037 \pm 1.932E-5

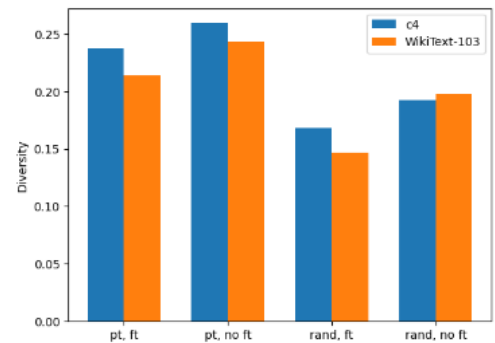
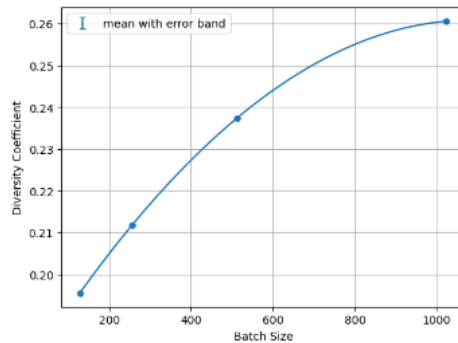
Cross Diversity coefficients of LLM pre-training datasets with 95% confidence intervals are 3-5 times higher than the conceptual lower bound and more than half that of the upper bound



Distribution of pairwise batch distances reflect conceptual and semantic dataset properties, therefore increasing trust in the diversity coefficient



Diversity coefficient of GINC datasets with varying number of latent concepts and vocab sizes shows the diversity coefficient behaves as expected



Diversity coefficients of C4 computed using different task batch sizes show positive and diminishing returns with increasing batch size (left). Diversity coefficients of C4 and WikiText-103 computed using different GPT-2 probe network configurations show that random networks underestimate diversity vs. pretrained networks, and non-finetuned networks overestimate diversity vs. finetuned networks (right).

VOCAB: (w/definition)

Diversity coefficient: A formal metric used to quantify the quality of pre-training data in the context of large language models (LLMs). It measures the diversity of

datasets by calculating the expected cosine distance between pairs of Task2Vec embeddings of batches. A higher diversity coefficient indicates a more diverse and informative dataset.

Generative IN-Context Learning dataset: Generative IN-Context Learning (GINC) datasets are mixtures of Hidden Markov Models (HMMs) with varying numbers of latent concepts. These datasets are employed to analyze and validate the diversity coefficient, particularly in understanding its behavior with respect to the number of latent concepts and vocabulary size.

Task2Vec: An embedding method used for sequence data, specifically designed to compute the diversity coefficient. It involves computing embeddings using the Fisher Information Matrix (FIM) derived from fine-tuning the final layer of a neural network. The resulting embeddings serve as unique fingerprints for batches, allowing the measurement of diversity in pre-training datasets.

Embeddings: Vector representations of data points in a high-dimensional space. In the context of the paper, embeddings are generated using the Task2Vec method, capturing information about batches in pre-training datasets. These embeddings play a crucial role in computing the diversity coefficient, reflecting the intrinsic variability of data batches.

Fisher Information Matrix: A matrix derived from tuning or fine-tuning the final layer of a neural network. In the context of Task2Vec, the diagonal entries of the FIM are interpreted as measures of information that parameters contain about the generative distribution. These entries are utilized to generate embeddings for batches in the dataset.

Cosine distance: A metric used to measure the angular similarity between vectors. In the paper, cosine distance is applied to calculate the diversity coefficient by assessing the expected cosine distance between pairs of Task2Vec embeddings of batches. Higher cosine distances indicate greater diversity between batches.

Hidden Markov Models: Hidden Markov Models (HMMs) are statistical models that assume an underlying hidden structure influencing observed data. In the context of GINC datasets, HMMs with varying latent concepts are used to simulate synthetic datasets for studying the behavior of the diversity coefficient with different numbers of latent concepts.

Cited references to follow up on

Longpre, S., Yauney, G., Reif, E., Lee, K., Roberts, A., Zoph, B., Zhou, D., Wei, J., Robinson, K., Mimno, D., and Ippolito, D. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity. *arXiv preprint arXiv:2305.13169*, 2023. URL <https://doi.org/10.48550/arXiv.2305.13169>.

Follow up Questions

How does the diversity coefficient approach contribute to advancing discussions on data quality, and in what ways does it provide a more nuanced understanding compared to traditional measures in the context of large language models?

The paper emphasizes the importance of considering data diversity and quality in the context of pre-training large language models. How might the findings impact the development and evaluation of future language models, particularly in terms of addressing biases and ensuring robust performance?

The experiments highlight the impact of varying batch size and network parameters on the diversity coefficient. What practical recommendations can be drawn from these results for researchers and practitioners aiming to optimize the pre-training process for large language models?

Notes

Introduction

- With the focus on model and dataset scaling, the effectiveness of scaling the token counts of models relies fundamentally on the quality and coverage of the pre-training data
- Data quality and coverage are often overlooked or discussed in vague or imprecise ways
- The paper proposes grounding the discussion of quality with the diversity coefficient
- They use the diversity coefficient to quantify the data quality of publicly available datasets
- The diversity coefficient is high for these datasets relative to conceptually well-motivated lower and upper bounds
- The paper shows the following
 1. The diversity coefficient increases as one concatenates more pre-training datasets of different sources
 2. The task embedding distances used in the diversity coefficient groups
 3. As the number of latent concepts increases, the diversity coefficient increases
 4. A larger, more diverse vocabulary leads to a higher diversity coefficient
- Key contributions include
 1. A paradigm shift beyond dataset scale to a data-centric machine learning perspective through a formal data quality metric – the diversity coefficient
 2. Advancing discussions on data quality by measuring an aspect of quality–data diversity–using the diversity coefficient
 3. Validating the diversity coefficient by demonstrating its interpretability and correlation with intuitive diversity properties aligned with human intuitions
 4. Demonstrating the high diversity of public datasets for LLM pre-training
 5. Studying properties of different parameters for computing the formal diversity and therefore providing practitioners with simpler ways to evaluate the diversity coefficient

Methods**Task2Vec Embeddings for Sequence Data**

- The Task2Vec diversity coefficient is used to compute the formal diversity of a dataset
- The first step is computing the embeddings, which is done according to the original Task2Vec method using the entries of the Fisher Information Matrix that result from tuning fine-tuning the final layer of a neural network
- The diagonal entries of the FIM can be interpreted as a measure of the information that a given parameter contains about the generative distribution
 - It serves as a unique fingerprint, or feature vector, for a batch, which defines a task distribution

Diversity Coefficient

- The Task2Vec diversity coefficient is calculated as the expected cosine distance d between pairs of Task2Vec embeddings of batches

Cross Diversity Coefficient

- The cross diversity coefficient computes the expected cosine distances of embeddings of batches by sampling a batch from the two data sets separately without mixing

Backbone Used and Further Explanation of the Diversity Coefficient

- By measuring the distance between FIMs, the diversity coefficient captures the average intrinsic variability of batches in the underlying data distribution as a proxy for data coverage or information contained in the dataset
- The dataset diversity reflects how different batches are from each other

Recipe for Establishing if a Diversity Coefficient is High via the Conceptual Lower and Upper Bounds

- To establish if a diversity coefficient is high or low, two conceptually well-motivated reference values are used: the lower and upper bounds
- The conceptual lower bound is measured on a dataset with probability concentrated on an arbitrary token
- Lower bound created with vocabulary size of 2, assigning probability weight to `<eos>` token and a randomly selected non-special token
- Probability weight for `<eos>`: $1/\{\text{GPT-2 vocab size}\}$, remaining weight for the non-special token
- Conceptual upper bound measured on a synthetic dataset with equal probability for all tokens in GPT-2 tokenizer vocabulary
- High or maximum diversity dataset consists of random sequences with no underlying order in semantics, formatting, etc

LLM Pre-training Datasets

- The publicly available language datasets used in the paper were outlined as follows
 - C4
 - WikiText-103
 - The Pile
 - Pile-CC
 - HackerNews
 - NIH ExPorter
 - PubMed Abstracts
 - USPTO Backgrounds

Experiments & Results

Diversity Coefficients of Pre-training Data shows LLMs are Pre-trained on Formally Highly Diverse Data

- The experiment evaluated the diversity coefficient of the eight public datasets and computed the diversity coefficient of two concatenated datasets
- Results in Table 2 show diversity coefficients for eight LLM pre-training datasets and their conceptually motivated lower and upper bounds
- Measured diversity coefficients for concatenation of various publicly available datasets are also presented in Table 2
- Key observations:
 - Pre-training datasets generally have diversity coefficients 3-5 times greater than the theoretical lower bound and, on average, half the upper bound
 - WikiText-103, C4, The Pile, and Pile-CC exhibit high diversity coefficients (0.21, 0.25)
 - Pile-CC has higher diversity than C4, suggesting a potentially more stringent preprocessing method for Pile-CC from the Common Crawl corpus
 - Three sub-datasets of The Pile (NIH ExPorter, PubMed Abstracts, USPTO) show relatively low diversity (0.15-0.17), about half of the upper bound (0.4), possibly due to their specialized fields
 - Pile-CC and HackerNews have higher diversity, likely attributed to their broad topic coverage
 - Pile-CC exhibits higher diversity, aligning with its heterogeneous content composition

Concatenation of Datasets of Different Sources Produces Higher Measured Diversity

- To show that the concatenation of different datasets produces high diversity, the paper measures the diversity coefficient of C4 plus WikiText-103, as well as the diversity coefficient of the five sub-datasets of The Pile in Table 2
- Key Observations:
 - Diversity coefficient for C4 and WikiText-103 concatenated dataset is 0.2711, approximately +0.03-0.05 higher than each individual dataset
 - Diversity coefficient for concatenation of five sub-datasets of The Pile is 0.2939 (Table 2), about +0.04-0.1 (Figure 1) higher than each individual dataset
 - Concatenation of five sub-datasets of The Pile achieves the highest diversity coefficient in Table 2
 - Increase in diversity results from higher pairwise Task2Vec distances between batches from different datasets (see Figure 1)
 - Diversity coefficient is an average of all pairwise Task2Vec distances, aligning with human intuition that combining data from heterogeneous sources increases overall diversity.

Distribution of Pairwise Batch Distances Reflects Conceptual and Semantic Dataset Information

- To increase confidence in the diversity coefficient as a diversity metric, the distributions of the Task2Vec distances used to compute the coefficient were studied
- In particular, the alignment of the grouping of these distances with human conceptual and semantic understanding was noted
- Analyzed Task2Vec (cosine) distances between batches from five sub-datasets of The Pile

- Compared distances within individual sub-datasets and across different sub-datasets, visualized in Figure 1
- Combined datasets show increased diversity coefficient compared to individual datasets
- Expect higher diversity for pairings of unrelated datasets than related datasets, observed in Figure 1 (right)
- Pairings of conceptually unrelated datasets in concatenated C4 and WikiText-103 show higher distances than individual datasets
- Concatenated sub-datasets of The Pile exhibit higher distances for unrelated datasets above the dotted line, while related datasets group below
- Pile-CC and HackerNews anticipated to cover diverse topics due to their web-scale nature, with highest individual diversities and increases when combined with other datasets
- Distances between Pile-CC and HackerNews batches are the lowest among pairwise distances of concatenated datasets above the diversity coefficient, aligning with human intuition
- Findings reinforce trust in the diversity coefficient as a metric, as it aligns with human intuition in interpreting Task2Vec distances

Diversity Coefficient Captures LLM Pre-training Data Distributional Properties

- To instill further confidence in the diversity coefficient, a correlation analysis with data distributional properties was performed on a synthetic dataset, GINC
- **Experiments:**
 - GINC datasets, mixtures of HMMs with 1-10,000 latent concepts, analyzed
 - Diversity coefficient variation plotted as latent concepts increase (Figure 2, top)
 - Curve fitted for GINC datasets with fixed vocabulary sizes of 50 and 150
 - Fixed latent concepts at 5 and 5000, plotted diversity coefficient against increasing vocabulary size (Figure 2, bottom)
 - Curve fitted for GINC datasets with 5 and 5000 latent concepts
- **Results:**
 - Diversity coefficient increases with a greater number of latent concepts (Figure 2, top)
 - Diminishing returns observed
 - High R2 values (0.952 and 0.898)
 - Diversity coefficient saturates as more latent concepts are added
 - Hypothesized due to marginal increases in variation from increased overlap
 - Diversity coefficient increases with larger vocabularies (Figure 2, bottom)
 - Exponential pace observed
 - High R2 values (0.993 and 0.984)
 - Hypothesis: Exponential growth due to scaling the number of tokens creating a more diverse dataset
 - Results indicate the diversity coefficient successfully captures different distributional sources of variation in the data

Using the Diversity Coefficient in Practice: Setting Batch Size and Network Parameters

- **Experiments:**

- Tested sensitivity of computed diversity coefficient to changes in batch size and probe network parameters
- Varied batch size and observed impact on diversity coefficient for C4 (200 batches, pretrained, fine-tuned GPT-2)
- Tested various probe network configurations for C4 and WikiText-103 diversity coefficient measurement
- **Results:**
 - Diversity coefficient increases with task batch size but with diminishing returns (Figure 3, left)
 - Diminishing returns due to greater coverage in tokens, topics, document formats, etc., between batches
 - Using a random probe network underestimates diversity compared to pretrained networks
 - Using a non fine-tuned network overestimates diversity
 - Trends in diversity coefficient estimation consistent across C4 and WikiText-103
- **Recommendations:**
 - Suggested using a batch size of 512 for faster computations and fewer memory issues
 - Proposed diversity coefficient computation using random and non fine-tuned networks for efficiency, saving computational costs
 - Acknowledged absolute diversity coefficient values may differ but consistency in network configuration is crucial
 - Further validation needed to determine if forgoing pre-trained and/or fine-tuned probe networks can produce robust embeddings as the original Task2Vec method

Related Work

- Existing diversity metrics focused on GAN-produced data, using precision- and recall-based frameworks
- Similar to Task2Vec, these metrics use embedding functions and argue that data quality is distinct from diversity in GANs
- In the context of LLMs, data diversity considered a subset of data quality, important for in-context learning
- Diversity metric sufficient to capture a crucial aspect of data quality, aiding coverage and task inclusion in test datasets
- Large LLMs robust to noise; high diversity preferred, and evidence suggests current open LLM datasets have this property
- A proposed diversity metric, Vendi Score, doesn't rely on embedding functions, but its benefits are unclear. Computationally more expensive than Task2Vec
- Vendi Score assumes a suitable similarity function, lacking guidance on data representation
- Data representation fundamental to machine learning success; deep learning effective for dataset/task embeddings
- Task2Vec's end-to-end approach learns effective embeddings, more general, flexible, and scalable than Vendi Score

- Leave detailed comparison with Vendi Score for future work

Discussion

- Extending and validating Task2Vec diversity coefficient for natural language data, confirming open LLMs pre-trained on formally diverse data
- Intuitive properties verified through experiments, instilling confidence in diversity coefficient method
- Conceptually motivated lower and upper bounds aid understanding of diversity coefficient magnitude
- Bounds apply to symbolic vocabulary sequence data; multi-modal embedding method can address this limitation
- Method doesn't rely on activations from an arbitrarily selected layer; diversity coefficient well-justified and extensively tested
- Deep learning representations suggested due to their success in various machine learning domains
- Need for a data representation acknowledged; deep learning representations and open-source pre-trained models recommended
- Explore random networks and models with no fine-tuning for accessibility
- Diversity coefficient deemed reliable and trustworthy, suggested for building quality diverse datasets for capable LLMs
- Relationship between pre-training data diversity and LLM evaluation test performance explored, showing a negative correlation between diversity and cross-entropy loss
- Positive relationship between diversity and model performance conjectured, but more extensive experiments needed
- Experiment challenges due to the expensive nature of pre-training large language models

Article #12 Notes: A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity

Source Title	A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity
Source citation (APA Format)	Longpre, S., Yauney, G., Reif, E., Lee, K., Roberts, A., Zoph, B., Zhou, D., Wei, J., Robinson, K., Mimno, D., & Ippolito, D. (2023). <i>A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity</i> . (arXiv:2305.13169). arXiv. https://arxiv.org/abs/2305.13169
Original URL	https://arxiv.org/abs/2305.13169
Source type	Journal Article
Keywords	Language Model Performance, Data Curation Choices, Temporal Misalignment, Toxicity and Quality Filters, Domain Composition Effects
#Tags	#llm, #bigdata, #nlp, #modelperformance, #pretraining
Summary of key points + notes (include methodology)	The study investigates the impact of data curation choices on large language model (LM) performance, emphasizing the lack of documentation in model development processes. Findings reveal that the age of the dataset, quality and toxicity filters, and domain composition significantly affect LM behavior. Notably, quality filtering increases toxic generation, while toxicity filtering trades toxic generations for reduced generalization. The study also addresses temporal misalignment, suggesting that the temporal properties of pretraining corpora are crucial, especially for larger models. Additionally, the impact of domain composition on downstream performance is explored, emphasizing the importance of diverse data sources. The study involves a comprehensive examination of 28 1.5B parameter LM models, spanning various data curation decisions. Pretraining datasets include Common Crawl (C4) and The Pile, with considerations for dataset age, domain filtering, and content filtering. Evaluation encompasses tasks related to toxicity identification, toxic generation, question-answering, and domain generalization. The study introduces measures like Temporal Degradation to assess the effects of pretraining misalignment. Filtering effects are analyzed using quality and toxicity filters, and the impact of

domain composition is evaluated through downstream performance on diverse tasks, shedding light on crucial aspects of responsible LM development.

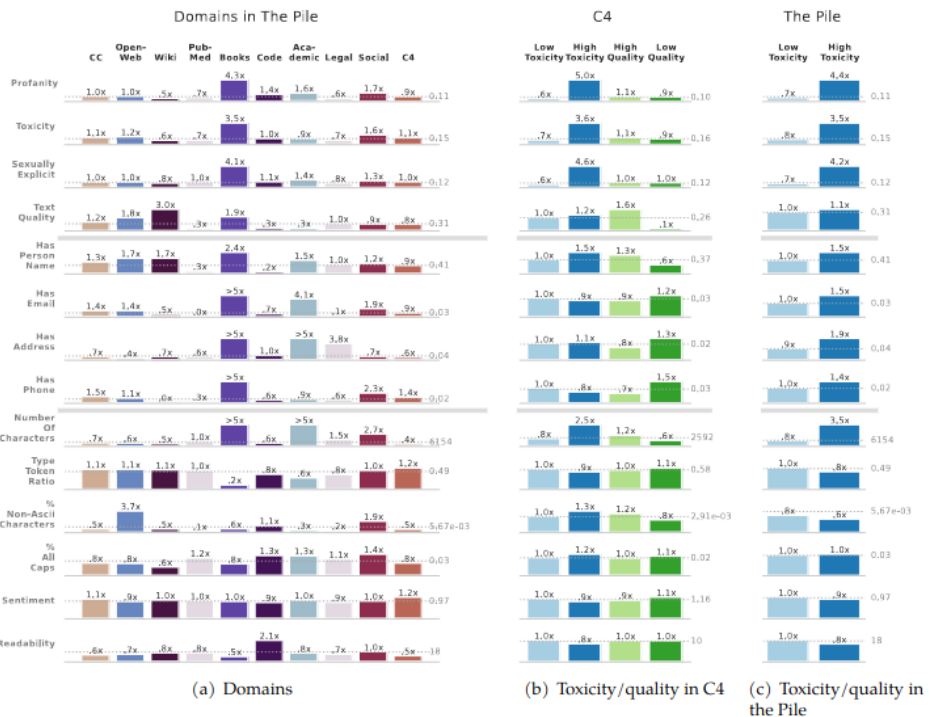
Research Question/Problem/
Need

What is the impact of data curation decisions on large language model performance, and how can this understanding contribute to responsible and effective model development?

Important Figures

MODEL	REPRESENTED DOMAINS (%)							PILE	C4	M-L	FILTERS		DATA	
	WIKI	WEB	BOOKS	DIALOG	CODE	ACAD	TOX				QUAL	PUB	YEAR	
BERT	76		24				X	X			H	Part	2018	
GPT-2		100					X	X			H	Part	2019	
RoBERTa	7	90	3				X	✓			H	Part	2019	
XLNet	8	89	3				X	✓			H	Part	2019	
T5	<1	99					X	✓			H	✓	2019	
GPT-3	3	82	16				X	✓	7%		C	X	2021	
GPT-J/NEO	1.5	38	15	4.5	13	28	✓	Part			C	✓	2020	
GLAM	6	46	20	28			X	✓			C	X	2021	
LAMDA	13	24		50	13		✓	✓	10%		C	X	2021	
ALPHACODE					100		X	X			H	X	2021	
CODEGEN	1	24	10	3	40	22	✓	Part			H	Part	2020	
CHINCHILLA	1	65	10		4		✓	✓			H	C	X	2021
MINERVA	<1	1.5	<1	2.5	<1	95	✓	✓	<1%		C	X	2022	
BLOOM	5	60	10	5	10	10	✓	✓	71%		H	C	Part	2021
PALM	4	28	13	50	5		X	✓	22%		C	X	2021	
GALACTICA	1	7	1		7	84	✓	Part			H	Part	2022	
LLAMA	4.5	82	4.5	2	4.5	2.5	Part	✓	4%		C	Part	2020	

A list of well-known language models and a quantitative breakdown of their pretraining data

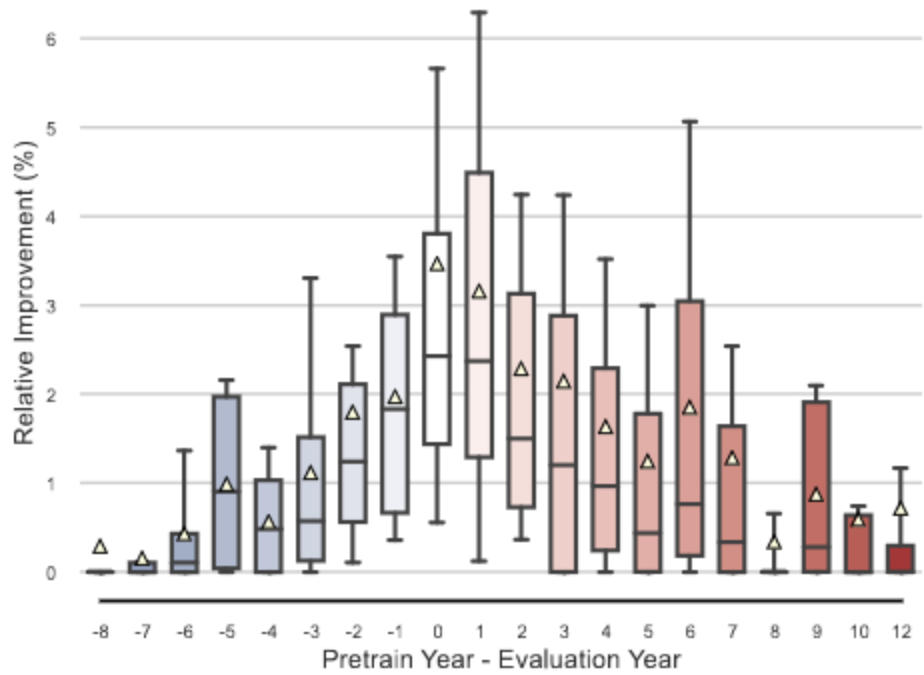


Feature differences across slices of the pretraining datasets

		PubCLS				NewSum				TwERC					
Pretrain Years	2013	78.9	79.2	78.5	75.1	23.6	32.1	27.7	17.9	85.0	84.9	84.9	82.7	83.1	83.5
	2016	76.8	76.7	79.0	76.3	23.3	32.0	27.9	16.1	85.2	84.8	85.9	83.1	83.4	83.4
	2019	75.0	76.3	77.1	73.2	22.8	31.2	27.4	18.1	84.6	84.4	84.6	84.0	83.8	84.6
	2022	74.0	75.7	76.8	73.4	22.7	31.2	27.4	17.8	82.7	83.7	84.4	82.9	82.7	83.6
		2010	2012	2014	2016	2010	2012	2014	2016	2014	2015	2016	2017	2018	2019

		AIC				PoliAff											
Pretrain Years	2013	98.2	98.0	95.0	91.3	94.4	88.7	82.7	89.0	91.2	71.2	70.8	74.7	71.5	82.0	82.2	74.9
	2016	98.1	98.2	95.2	93.1	95.1	88.5	80.4	88.1	90.6	70.9	72.3	75.8	72.6	82.2	82.4	76.0
	2019	97.8	98.7	93.9	93.4	96.0	90.5	80.2	87.8	90.7	70.4	72.0	75.8	73.4	83.1	82.8	75.9
	2022	97.6	98.4	94.4	91.4	95.1	89.0	79.4	87.1	89.4	70.8	71.4	75.0	71.0	82.5	83.1	76.8
		2014	2015	2016	2017	2018	2019	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021

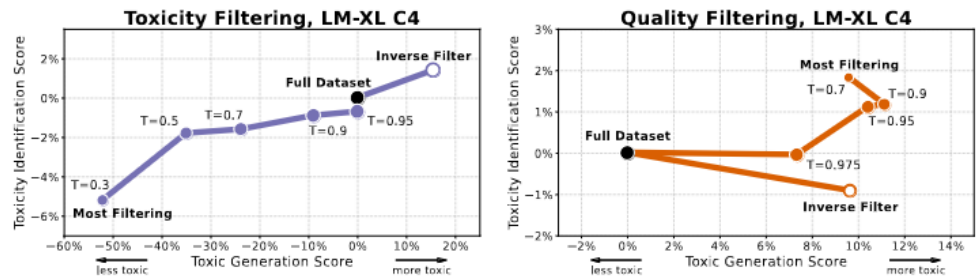
Temporal Misalignment between Pretraining and Evaluation causes performance degradation



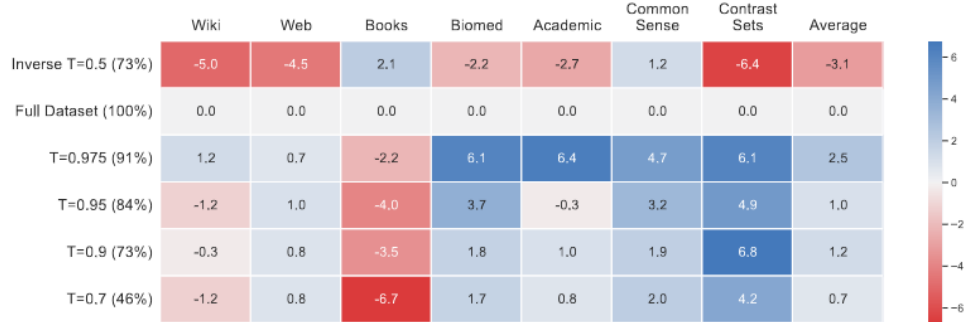
The mean relative performance over 5 datasets (y-axis) increases as temporal misalignment (x-axis) approaches zero

DOMAIN	TASK	FINETUNING				PRETRAINING			
		LM-SMALL		LM-XL		LM-SMALL		LM-XL	
		TD	r	TD	r	TD	r	TD	r
NEWS	PUBCLS	5.82	0.84	5.63	0.80	0.02	0.01 [†]	0.59	0.67
	NEWSUM	0.80	0.82	2.91	0.92	-0.31	-0.29	0.73	0.45
TWITTER	POLIAFF	3.74	0.84	4.93	0.89	0.50	0.21	0.28	0.56
	TwiERC	0.49	0.73	0.53	0.82	0.05	0.27	0.23	0.72
SCIENCE	AIC	0.94	0.83	0.24	0.36	0.11	0.18 [†]	0.23	0.66
MEAN		2.36	0.81	2.84	0.76	0.08	0.07	0.41	0.61

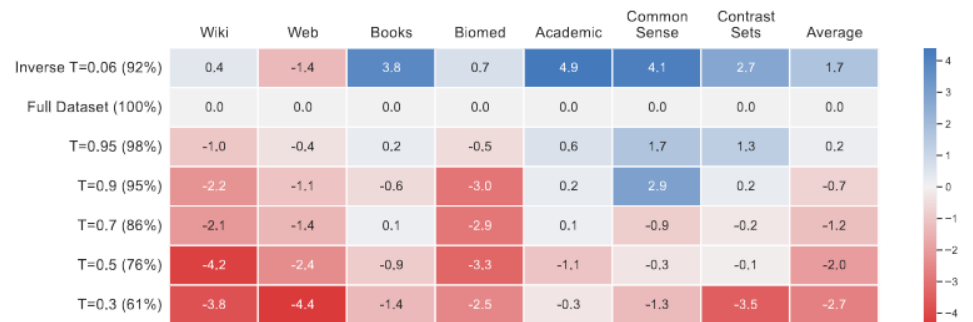
Temporal Degradation due to pretraining is significant and persistent across domains



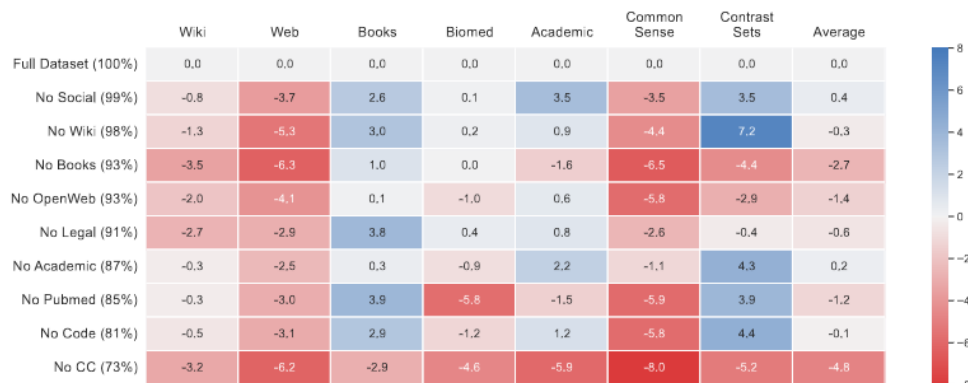
Toxicity filtering the pretraining dataset decreases the ability of LM-XL to identify toxicity and to generate toxic text



Quality filtering C4 increases LM-XL's downstream performance on all QA task domains, except for Books



Toxicity filtering C4 reduces LM-XL's downstream performance on most QA task domains



QA tasks are affected by removing domains when pretraining LM-XL

FILTER	% DATA	TOXICITY IDENTIFICATION (↑)					TOXIC GENERATION (↓)			
		SBF	Toxigen	DH R3	DH R4	Score	RTP-T	RTP-NT	RepBias	Score
FULL DATASET	100.0	90.7	90.8	88.7	84.1	0.0	88.9	45.4	4.6±0.7	0.0
NO SOCIAL	98.8	90.9	91.0	87.8	84.9	+0.1	85.4	47.2	4.7±0.8	+0.4
NO WIKI	97.9	90.6	90.8	88.1	83.6	-0.4	89.0	49.4	4.8±0.6	+4.2
NO BOOKS	93.1	89.9	90.3	87.1	82.6	-1.3	87.4	43.5	4.0±0.8	-6.2
NO OPENWEB	93.1	89.9	90.3	86.4	82.5	-1.5	88.0	42.1	4.3±0.6	-5.2
NO LEGAL	91.0	90.9	90.8	88.1	83.0	-0.4	88.2	46.1	4.7±0.8	+0.8
NO ACADEMIC	87.1	90.7	91.0	88.2	84.5	+0.0	86.5	46.4	4.5±0.7	-1.2
NO PUBMED	85.1	90.6	90.8	88.0	84.3	-0.2	87.6	46.3	4.6±0.7	-0.2
NO CODE	80.9	91.0	91.2	88.5	84.5	+0.2	87.6	46.5	4.7±0.7	+0.6
NO CC	73.1	89.9	90.0	85.3	82.4	-1.9	87.8	46.2	4.3±0.6	-2.1

Effect of the Pile's domain composition on toxicity identification and generation

VOCAB: (w/definition)

Toxicity: The presence of harmful or offensive content, such as profanity, insults, or threats, within textual data, evaluated using methods like the Perspective API to assign toxicity scores

Domain compositions: The diverse sources or categories from which data is drawn, influencing the makeup of the pretraining dataset and subsequently affecting the behavior of language models in different domains

Heuristic filtering: The use of practical and experience-based rules or methods to filter and curate datasets, often applied to identify and remove undesirable or low-quality content

Rule-based classifiers: Algorithms or systems that classify data into predefined categories based on explicit rules, frequently used for categorizing documents into quality or toxicity levels in the context of language models

N-gram filter: A filtering technique based on the analysis of contiguous sequences of n-grams (sets of n adjacent words), often employed to identify or eliminate specific patterns or content in language datasets

	<p>PII: Sensitive information that can be used to identify individuals, such as personal names, addresses, and emails, which is often removed or handled with care during data curation</p> <p>Temporal misalignment: A discrepancy between the time of dataset pretraining and the time of evaluation, impacting language model performance due to changes in language use over time</p> <p>Temporal Degradation (TD): A measure introduced to assess the decline in performance resulting from the time gap between pretraining and evaluation, particularly relevant for larger language models</p> <p>Pearson correlation: A statistical measure indicating the strength and direction of a linear relationship between two variables, employed to analyze the correlation between temporal misalignment and performance degradation</p> <p>Wald test: A statistical test used to assess the significance of coefficients in regression analysis, potentially applied to evaluate the significance of factors influencing language model performance</p> <p>SafeSearch filters: Filtering mechanisms designed to block or restrict access to content that may be considered inappropriate, often used to improve the quality of web-derived training data</p> <p>Safety discriminators: Mechanisms or filters integrated into language models to discern and mitigate unsafe or undesirable outputs, enhancing the model's safety during generation</p> <p>Presentist bias: The potential bias introduced when language models are predominantly trained on recent data, leading to a skewed representation of language that may not align with historical language use</p> <p>Ablating: The process of selectively removing or excluding specific components, in this context, certain domains or data sources during the analysis of language model behavior and performance</p>
Cited references to follow up on	<p>Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, et al. Quality at a glance: An audit of web-crawled multilingual datasets. <i>Transactions of the Association for Computational Linguistics</i>, 10:50–72, 2022.</p>
Follow up Questions	<p>How might the findings of this study influence the development and deployment of large language models, especially in industries where responsible data use and ethical considerations are paramount?</p> <p>Given the observed trade-offs in toxicity and quality filtering, what strategies or tools can model developers employ to strike a balance between mitigating toxic</p>

content generation and maintaining model generalization across diverse datasets?

In light of the temporal misalignment challenges highlighted, what are potential solutions or recommendations for adapting language model training practices to better align with evolving language use over time, particularly for larger models that are more susceptible to degradation?

Notes (written with the assistance of ChatGPT)

Introduction

- Modern LMs' performance relies on self-supervised pretraining on massive text datasets
- Model developers decide dataset composition, filtering, and document collection protocols, often undocumented
- Lack of documentation hinders responsible data use and effective model development
- This study systematically tests common data design decisions' impact on model performance
- Findings and recommendations presented for model developers
- Age of dataset affects performance, leading to degradation if evaluation data is before or after pretraining data collection
- Quality and toxicity filters have significant but opposite effects on model behavior
- Quality filtering increases toxic generation and downstream performance; toxicity filtering trades toxic generations for reduced generalization
- Inverse toxicity filters demonstrate targeted benefits
- Domain composition influences performance, with high-quality and heterogeneous data contributing to toxic generation
- Benefits of training on diverse data often greater than collecting targeted domain-specific data
- Best-performing models use all data sources, recommending practitioners to include diverse sources
- Experiments constitute the largest publicly documented LM data curation study, spanning 28 1.5B parameter models
- Findings justify computational cost and inform model developers training the next wave of LMs

Methodology

Pretraining Datasets

- The two datasets used in the study are
 - C4
 - The Pile

Data Curation Choices

- Dataset Age
 - New versions of C4 are created by regenerating snapshots of the Common Crawl from different years
 - Multiple time-based collections are not available for the Pile
- Domain Filtering
 - Both C4 and the Pile draw from multiple distinct data sources
 - The Pile explicitly delineates 22 distinct sources from web pages, Wikipedia articles, code repositories, online forums, legal texts, and research paper archives

- Documents from different domains are selectively removed to control for the topical content of the pretraining collection
- Content Filtering
 - Datasets from weakly curated internet sources tend to contain low-quality, toxic, or offensive content
 - Various approaches to determining document appropriateness, including negatively-defined filters (removing specific categories), positively-defined filters (keeping specific categories), and features-based filters
 - Evaluation of the impact of two document-level, classifier-based filters for toxic content and quality content
- Quality Filters
 - Language models create quality classifiers to distinguish between "high-quality" corpora and other documents
 - The classifier assigns each document a score from 0 (high quality) to 1 (low quality)
 - Experimentation with removing documents above and below specified quality thresholds
- Toxicity Filters
 - The Perspective API is used to identify toxic content, assigning toxicity scores based on profanity, identity-based negativity, insults, or threats
 - Experimentation with removing documents above and below specified toxicity thresholds
 - In addition to the classifier-based filter, experimentation with an n-gram based filter used in the original version of the C4 dataset

Evaluation

- Measurement of effects of time, topic, and toxicity on pretrained models
- English-language tasks include toxicity identification, toxic generation, question-answering tasks from diverse domains, and tasks with temporal annotations
- Comparison of general utility and performance on tasks influenced by dataset characteristics
- Finetuning of each pretrained model on relevant datasets for downstream tasks, evaluated on the same testing data
- Downstream Task Performance
 - Evaluation of each pretrained model's performance on downstream tasks, attributing systematic differences to pretraining variations
 - Reporting mean performance relative to a baseline, often models trained on an unfiltered dataset
- Domain Generalization Evaluation
 - Assessment on the union of two question-answering benchmarks: Machine Reading for Question Answering (MRQA) and UnifiedQA
 - 30 unique QA datasets spanning diverse domains, measuring the impact of topic alignment
- Temporal Misalignment Evaluation

- Investigation of the impact of dataset collection time on downstream model abilities
- Evaluation on five datasets with varying domains to explore potential performance variations between pretraining and evaluation time
- Toxic Generation Evaluation
 - Assessment of language model behavior in generating profane, sexually explicit, insulting, or obscene text
 - Evaluation using prompts designed to elicit biased or toxic outputs related to gender, race, and religion
 - Measurement of the fraction of generated continuations assigned a high toxicity score by the Perspective API
 - Utilization of the RealToxicityPrompts dataset, consisting of labeled toxic text excerpts
- Toxicity Identification Evaluation
 - Assessment of the ability of language models to recognize toxic language
 - Importance for content moderation on communication platforms
 - Evaluation with various toxicity interpretations using train and test sets from Social Bias Frames (SBF), DynaHate (DH), and Toxigen

Models

- Two sizes of decoder-only transformer, Transformer-based language models were used
 - LM-XL: 1.5B parameters
 - LM-SMALL: 20M parameters

Impact of Data Curation on Data Characteristics

- Observational Findings
 - The Pile's documents exhibit differences compared to C4
 - Longer, more readable, higher quality, but contain more personally identifiable information (PII)
 - Books domain is an outlier with
 - Longest, most readable, most toxic, and most PII-filled documents
 - Contains high-quality text
 - High toxicity and low-quality documents show similarities in high PII amounts but differ in average length and quality
 - More recent web-scraped text is more diverse and less toxic but also lower quality
- Data Analysis
 - Calculation of features for each document, including toxicity and quality metrics, PII categories, and text statistics
 - Substantial interactions between curation choices impacting data features
 - Features include average word length, readability, type-token ratio, sentiment, and more (details in Appendix D)
- C4 vs. Pile
 - Pile documents compared to C4

- Longer (2.4x), more non-ASCII characters (1.9x), higher quality (1.2x), more readable (1.8x)
 - Contain more PII, particularly personal names, addresses, and emails
- Toxicity and Quality Relationship
 - High toxicity does not necessarily correlate with low quality
 - High toxicity documents exhibit higher text quality than low toxicity documents
 - Little discernible difference in feature measurements for profanity, toxicity, and sexually explicit content between low vs. high-quality content
- Domain Characteristics
 - Books domain stands out with more profane, toxic, and sexual content, yet greater predicted quality
 - High toxicity documents in both C4 and Pile are longer, more profane, sexually explicit, and toxic
 - Pile documents with high toxicity are more likely to have various kinds of PII
 - OpenWeb provides the most lexical and linguistic diversity, Wikipedia has the highest quality text, technical domains score low on predicted quality
- Temporal Trends in C4
 - Increase in non-ASCII characters over recent years
 - Decline in measured text quality
 - Slight decrease in toxicity scores and increase in sentiment over time

Impact of Dataset Age on Pretrained Models

- Temporal Misalignment Findings
 - Both models and evaluation datasets become stale over time
 - Temporal misalignment persists even after finetuning
 - Effects of pretraining misalignment are more pronounced in larger models
- Observations and Context
 - Models are frequently updated with new finetuning data, but pretraining is expensive
 - Majority of downloaded models are static and rarely updated, constituting ~58% of all downloads on HuggingFace
 - Language use changes over time, and temporal misalignment between finetuning and evaluation datasets leads to performance degradation
- Experimental Setup
 - Pretrained four autoregressive language models on different C4 versions (2013, 2016, 2019, 2022)
 - Removed data scraped after the cutoff year
 - Evaluated impact of pretraining time on NLP using tasks split by year in News, Twitter, and Science domains
- Temporal Degradation Metrics
 - Replicated performance degradation observed by Luu et al. (2021)
 - Introduced Temporal Degradation (TD) measure for pretraining time and evaluation time differences

- High correlation (average Pearson correlation of 0.61) between pretraining temporal misalignment and performance degradation
- Pretraining misalignment effects are non-trivial (0.4 on average for one year difference)
- Effect on Model Performance
 - Pretraining misalignment not overcome by significant finetuning
 - Asymmetric effects observed, with degradation steeper when evaluation year is after pretraining year
 - Models and evaluations become stale, leading to performance differences between older and newer models on respective evaluations
- Temporal Degradation Across Model Sizes
 - Greater temporal degradation for larger models (LM-XL with 1.5B parameters) compared to smaller models (LM-Small with 20M parameters)

Impact of Quality & Toxicity Filters on Pretrained Models

- Filtering Effects Findings
 - Quality and toxicity filters have distinct impacts on model performance
 - Quality filters significantly enhance performance despite reducing training data
 - Effects of quality filtering are not easily predictable based on dataset characteristics
 - Toxicity filtering involves a trade-off between generalization, toxicity identification, and risk of toxic generation
 - For toxicity identification tasks, an inverse toxicity filter is recommended
- Common Filter Usage in Language Models
 - Modern large language models use quality and/or toxicity filters in pretraining datasets
 - Examples include n-gram filters, SafeSearch filters, and "safety discriminators"
- Experimental Setup and Implementation of Filters
 - Implemented quality and toxicity filters at various thresholds on Pile and C4 datasets
 - Varied the quantity of toxic and low-quality text during pretraining
- Quality Filters Impact on Performance
 - Quality filters significantly improve performance across various tasks
 - Improvements observed in toxicity identification (2%) and most QA task categories (1-6%)
 - Greater quality filtering outperforms the unfiltered baseline, even with 10%+ data removal
 - Notable performance improvements persist beyond $T = 0.975$ for QA tasks
- Dataset Quality Characteristics and Filtering Effects
 - Dataset quality characteristics are not strong indicators of filtering effects
 - QA tasks in Books, Wikipedia, and Web categories benefit less from quality filtering
 - Academic and biomedical data, ranked lower in quality, benefit the most from quality filtering
- Challenges with One-Dimensional Quality Measurement
 - Optimizing on one quality measure is insufficient to predict or improve performance across domains

- Wikipedia and Web QA tasks are negatively affected by the inverse filter
- Both quality and inverse quality filters lead to models with higher toxic generation tendencies
- Toxicity Filtering Trade-offs
 - Toxicity filtering involves a trade-off between toxic identification and toxic generation goals
 - Models from heavily filtered datasets have less toxic generation but poorer toxicity identification
 - Inverse toxicity filter performs best for toxicity identification across all datasets
 - Filtering strategy should align with the intended behavior of the model

Impact of Domain Composition on Pretrained Models

- Impact of Pretraining Source Domains on Downstream Performance
 - Common Crawl, OpenWeb, and Books have the most positive impact on downstream performance
 - Data source heterogeneity is more crucial than data quality or size
 - Inclusion of as many pretraining data sources as possible is beneficial
- Experimental Setup
 - Grouped Pile data sources into nine domains
 - Pretrained LM-XL with the full dataset minus each domain
 - Evaluated downstream performance on 27 QA tasks from MRQA and UnifiedQA
 - Domains: Common Crawl (CC), OpenWeb, Wikipedia, Books, PubMed, Academic, Code & Math, Legal, and Social
- Key Findings
 - Web-based domains like CC, Books, and OpenWeb have the strongest positive effects on performance
 - Heterogeneity and quality are more important factors than the quantity of data
 - Domain heterogeneity is often more beneficial than targeted data, even for targeted evaluations
 - Best-performing models utilize all pretraining data sources, except for targeted domains like Code and Academic
 - Web and Books domains present a trade-off between toxic identification and generation

Discussion

- Guided by Intuition
 - Pretraining dataset curation often guided by intuitions, lacking thorough evaluation
 - Documentation debt is maintained, limiting knowledge sharing
- Impact of Curation Choices
 - Pretraining curation choices significantly impact models and cannot be fully mitigated by finetuning
 - Dataset curation policies should be treated as hyperparameters
 - Better tools are needed to model the relationship between data and model capabilities

- Age of Pretraining Corpus
 - Staleness of pretraining data affects model performance, especially for larger, more capable models
 - Finetuning on newer data may introduce a "presentist" bias, and the effect of staleness is not overcome by ample finetuning data
 - Temporal properties of pretraining corpora are crucial for larger models and novel tasks
- Recommendations
 - Model creators should report the temporal distribution of pretraining data
 - Users need awareness of potential performance degradations on newer datasets
- Data Source Composition
 - Composition of the pretraining corpus significantly impacts downstream performance
 - Ablating any data source in diverse corpora like the Pile negatively affects generalization to text-to-text tasks
 - Future work should focus on collecting more diverse web and books content
- Filtering for Toxicity and Quality
 - Filtering for toxicity and quality involves normative decisions that modify dataset bias
 - There's a trade-off between a model's generalization abilities and its tendency to generate toxic content
 - Toxic identification should be prioritized over curbing toxic generation during pretraining
 - Quality filters, despite removing large portions of training data, significantly improve performance across domains

Limitations

- Largest publicly documented LM pretraining data ablation study
- Spans 28 1.5B parameter models, surpassing other studies like GLaM, miniBertAs, MultiBertAs, and Pythia
- Computationally and environmentally costly experiments
- Carefully curated experiments focused on age of corpora, quality filters, toxicity filters, and source domains
- Limited multiple rounds of reflection and repetition, striking a balance between computational costs and reproducible validity
- Use of Perspective's API for toxicity evaluation, with limitations on irreproducibility
- English datasets used, highlighting the importance of considering training composition for multilingual and non-English models
- Focus on finetuned settings rather than zero- or few-shot prompting
- Findings may be correlated to prompted settings but not explicitly established

Related Work

- General-purpose models include ELMO, BERT, BERT's descendants, XLNet, T5, GPT-2, GPT-Neo, OPT, LLaMA, Pythia, and BLOOM
- Publicly available models and proprietary models by companies like Alphabet and OpenAI

- Filtering techniques used to improve web-derived training data quality, including classifiers, SafeSearch, and heuristics
- Pretraining data analysis studies by Dodge et al., Luccioni and Viviano, Kreutzer et al., Lee et al., Kaddour, and Zhao et al.
- Mixed findings on the effects of data detoxification techniques, such as data filters, on underrepresented communities
- Instruction tuning and alignment tuning as methods to reduce unwanted toxic generation
- Language's distribution shift over time and its impact on model performance on new test sets
- Proposed remedies for temporal degradation include finetuning on more recent data, adaptive/continuous pretraining, data augmentation, and modeling text with timestamps
- Exhaustion of high-quality text data on the web for training large language models
- Adaptation strategies for pretrained models to new downstream domains, including domain adaptive pretraining, intermediate finetuning tasks, balancing data sources, data selection, augmentation, and active learning
- Research on rebalancing mixtures of datasets, importance sampling for selecting useful subsets, and benchmarking effects of intermediate finetuning tasks
- Exploration of scaling model size, pretraining data amount, and number of pretraining steps
- Investigation into how temporal pretraining misalignment varies on different model sizes

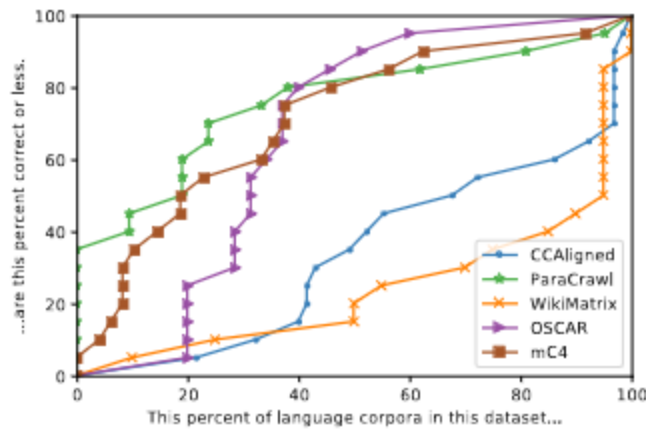
Article #13 Notes: Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets

Source Title	Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets
Source citation (APA Format)	Kreutzer, J., Caswell, I., Wang, L., Wahab, A., van Esch, D., Ulzii-Orshikh, N., Tapo, A., Subramani, N., Sokolov, A., Sikasote, C., Setyawan, M., Sarin, S., Samb, S., Sagot, B., Rivera, C., Rios, A., Papadimitriou, I., Osei, S., Suarez, P. O., ... Adeyemi, M. (2021). <i>Quality at a glance: An audit of web-crawled multilingual datasets</i> . (arXiv:2103.12028). arXiv. https://arxiv.org/abs/2103.12028
Original URL	https://arxiv.org/abs/2103.12028
Source type	Journal Article
Keywords	Multilingual datasets, NLP research, Data quality, Dataset mislabeling, Downstream applications
#Tags	#llm, #bigdata, #nlp, #dataquality, #multilingualdata
Summary of key points + notes (include methodology)	The research focuses on enhancing access to multilingual datasets for NLP, emphasizing web-derived collections like ParaCrawl, WikiMatrix, CCAIined, OSCAR, etc. The study enabled the development of highly multilingual models but highlighted the lower quality of automatically crawled datasets compared to hand-curated collections. A manual data audit involving 230 per-language subsets revealed significant challenges, leading to proposed solutions for effective, low-effort auditing and an error taxonomy. The analysis uncovered issues like mislabeled language codes, nonlinguistic content, and offensive material. The study advocates for improved data quality through techniques such as automatic filtering and emphasizes the importance of standardized language codes. The research's methodology involved a diverse group of 51 non-expert participants from the NLP community, who manually annotated a random sample of 100 lines per language in each dataset. The audit results exposed severe quality issues across datasets, especially for lower-resource languages, leading to valuable recommendations for the NLP community's future data releases and evaluations.
Research Question/Problem/	How can the quality and reliability of multilingual datasets for natural language

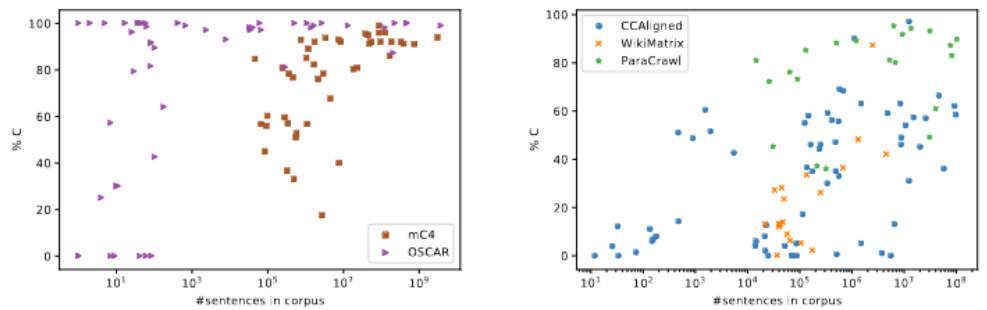
Need	processing be enhanced, particularly addressing challenges such as mislabeled language codes and low-quality content, to improve the effectiveness of NLP research and applications?																																																																																									
Important Figures	<table border="1" data-bbox="532 359 1503 625"> <thead> <tr> <th rowspan="2"></th> <th colspan="3">Parallel</th> <th colspan="2">Monolingual</th> </tr> <tr> <th>CCAligned</th> <th>ParaCrawl v7.1</th> <th>WikiMatrix</th> <th>OSCAR</th> <th>mC4</th> </tr> </thead> <tbody> <tr> <td>#languages</td> <td>137</td> <td>41</td> <td>85</td> <td>166</td> <td>101</td> </tr> <tr> <td>Source</td> <td>CC 2013–2020</td> <td>selected websites</td> <td>Wikipedia</td> <td>CC 11/2018</td> <td>CC all</td> </tr> <tr> <td>Filtering level</td> <td>document</td> <td>sentence</td> <td>sentence</td> <td>document</td> <td>document</td> </tr> <tr> <td>Langid</td> <td>FastText</td> <td>CLD2</td> <td>FastText</td> <td>FastText</td> <td>CLD3</td> </tr> <tr> <td>Alignment</td> <td>LASER</td> <td>Vec/Hun/BLEU-Align</td> <td>LASER</td> <td>-</td> <td>-</td> </tr> <tr> <td>Evaluation</td> <td>TED-6</td> <td>WMT-5</td> <td>TED-45</td> <td>POS/DEP-5</td> <td>XTREME</td> </tr> </tbody> </table> <p data-bbox="521 653 1479 720"><i>Comparison of parallel and monolingual corpora extracted from web documents, including their downstream evaluation task</i></p> <table border="1" data-bbox="532 779 1503 1440"> <thead> <tr> <th colspan="2">Correct Codes</th> </tr> </thead> <tbody> <tr> <td>C: <i>Correct translation, any</i></td> <td>Combined label for CC, CB, CS</td> </tr> <tr> <td>CC: <i>Correct translation, natural sentence</i></td> <td></td> </tr> <tr> <td>en The Constitution of South Africa</td> <td>nso Molaotheo wa Rephabliki ya Afrika Borwa</td> </tr> <tr> <td>en Transforming your swimming pool into a pond</td> <td>de Umbau Ihres Swimmingpools zum Teich</td> </tr> <tr> <td>CB: <i>Correct translation, Boilerplate or low quality</i></td> <td></td> </tr> <tr> <td>en Reference number: 13634</td> <td>ln Motango ya référéncé: 13634</td> </tr> <tr> <td>en Latest Smell Stop Articles</td> <td>fil Pinakabagong mga Artikulo Smell Stop</td> </tr> <tr> <td>CS: <i>Correct translation, Short</i></td> <td></td> </tr> <tr> <td>en movies, dad</td> <td>it cinema, papà</td> </tr> <tr> <td>en Halloween - without me</td> <td>ay Halloween – janiw nayampejj</td> </tr> <tr> <th colspan="2">Error Codes</th> </tr> <tr> <td>X: <i>Incorrect translation, but both correct languages</i></td> <td></td> </tr> <tr> <td>en A map of the arrondissements of Paris</td> <td>kg Paris kele mbanza ya kimfumu ya Fwalansa.</td> </tr> <tr> <td>en Ask a question</td> <td>tr Soru sor Kullanıma göre seçim</td> </tr> <tr> <td>WL: <i>Source OR target wrong language, but both still linguistic content</i></td> <td></td> </tr> <tr> <td>en The ISO3 language code is zho</td> <td>zza Táim eadra brachach mar bhionns na frogannaidhe.</td> </tr> <tr> <td>en Der Werwolf — sprach der gute Mann,</td> <td>de des Weswolfs, Genitiv sodann,</td> </tr> <tr> <td>NL: <i>Not a language: at least one of source and target are not linguistic content</i></td> <td></td> </tr> <tr> <td>en EntryScan 4 _</td> <td>tn TSA PM704 _</td> </tr> <tr> <td>en organic peanut butter</td> <td>ckb 🍪🍪🍪🍪🍪🍪</td> </tr> </tbody> </table> <p data-bbox="521 1461 1268 1493"><i>Annotation codes for parallel data with sentence pair examples</i></p>		Parallel			Monolingual		CCAligned	ParaCrawl v7.1	WikiMatrix	OSCAR	mC4	#languages	137	41	85	166	101	Source	CC 2013–2020	selected websites	Wikipedia	CC 11/2018	CC all	Filtering level	document	sentence	sentence	document	document	Langid	FastText	CLD2	FastText	FastText	CLD3	Alignment	LASER	Vec/Hun/BLEU-Align	LASER	-	-	Evaluation	TED-6	WMT-5	TED-45	POS/DEP-5	XTREME	Correct Codes		C: <i>Correct translation, any</i>	Combined label for CC, CB, CS	CC: <i>Correct translation, natural sentence</i>		en The Constitution of South Africa	nso Molaotheo wa Rephabliki ya Afrika Borwa	en Transforming your swimming pool into a pond	de Umbau Ihres Swimmingpools zum Teich	CB: <i>Correct translation, Boilerplate or low quality</i>		en Reference number: 13634	ln Motango ya référéncé: 13634	en Latest Smell Stop Articles	fil Pinakabagong mga Artikulo Smell Stop	CS: <i>Correct translation, Short</i>		en movies, dad	it cinema, papà	en Halloween - without me	ay Halloween – janiw nayampejj	Error Codes		X: <i>Incorrect translation, but both correct languages</i>		en A map of the arrondissements of Paris	kg Paris kele mbanza ya kimfumu ya Fwalansa.	en Ask a question	tr Soru sor Kullanıma göre seçim	WL: <i>Source OR target wrong language, but both still linguistic content</i>		en The ISO3 language code is zho	zza Táim eadra brachach mar bhionns na frogannaidhe.	en Der Werwolf — sprach der gute Mann,	de des Weswolfs, Genitiv sodann,	NL: <i>Not a language: at least one of source and target are not linguistic content</i>		en EntryScan 4 _	tn TSA PM704 _	en organic peanut butter	ckb 🍪🍪🍪🍪🍪🍪
	Parallel			Monolingual																																																																																						
	CCAligned	ParaCrawl v7.1	WikiMatrix	OSCAR	mC4																																																																																					
#languages	137	41	85	166	101																																																																																					
Source	CC 2013–2020	selected websites	Wikipedia	CC 11/2018	CC all																																																																																					
Filtering level	document	sentence	sentence	document	document																																																																																					
Langid	FastText	CLD2	FastText	FastText	CLD3																																																																																					
Alignment	LASER	Vec/Hun/BLEU-Align	LASER	-	-																																																																																					
Evaluation	TED-6	WMT-5	TED-45	POS/DEP-5	XTREME																																																																																					
Correct Codes																																																																																										
C: <i>Correct translation, any</i>	Combined label for CC, CB, CS																																																																																									
CC: <i>Correct translation, natural sentence</i>																																																																																										
en The Constitution of South Africa	nso Molaotheo wa Rephabliki ya Afrika Borwa																																																																																									
en Transforming your swimming pool into a pond	de Umbau Ihres Swimmingpools zum Teich																																																																																									
CB: <i>Correct translation, Boilerplate or low quality</i>																																																																																										
en Reference number: 13634	ln Motango ya référéncé: 13634																																																																																									
en Latest Smell Stop Articles	fil Pinakabagong mga Artikulo Smell Stop																																																																																									
CS: <i>Correct translation, Short</i>																																																																																										
en movies, dad	it cinema, papà																																																																																									
en Halloween - without me	ay Halloween – janiw nayampejj																																																																																									
Error Codes																																																																																										
X: <i>Incorrect translation, but both correct languages</i>																																																																																										
en A map of the arrondissements of Paris	kg Paris kele mbanza ya kimfumu ya Fwalansa.																																																																																									
en Ask a question	tr Soru sor Kullanıma göre seçim																																																																																									
WL: <i>Source OR target wrong language, but both still linguistic content</i>																																																																																										
en The ISO3 language code is zho	zza Táim eadra brachach mar bhionns na frogannaidhe.																																																																																									
en Der Werwolf — sprach der gute Mann,	de des Weswolfs, Genitiv sodann,																																																																																									
NL: <i>Not a language: at least one of source and target are not linguistic content</i>																																																																																										
en EntryScan 4 _	tn TSA PM704 _																																																																																									
en organic peanut butter	ckb 🍪🍪🍪🍪🍪🍪																																																																																									

		Parallel			Monolingual	
		CCAligned	ParaCrawl v7.1	WikiMatrix	OSCAR	mC4
#langs audited / total		65 / 119	21 / 38	20 / 78	51 / 166	48 / 108
%langs audited		54.62%	55.26%	25.64%	30.72%	44.44%
#sents audited / total		8037 / 907M	2214 / 521M	1997 / 95M	3517 / 8.4B	5314 / 8.5B
%sents audited		0.00089%	0.00043%	0.00211%	0.00004%	0.00006%
macro	C	29.25%	76.14%	23.74%	87.21%	72.40%
	X	29.46%	19.17%	68.18%	-	-
	WL	9.44%	3.43%	6.08%	6.26%	15.98%
	NL	31.42%	1.13%	1.60%	6.54%	11.40%
	offensive	0.01%	0.00%	0.00%	0.14%	0.06%
	porn	5.30%	0.63%	0.00%	0.48%	0.36%
micro	C	53.52%	83.00%	50.58%	98.72%	92.66%
	X	32.25%	15.27%	47.10%	-	-
	WL	3.60%	1.04%	1.35%	0.52%	2.33%
	NL	10.53%	0.69%	0.94%	0.75%	5.01%
	offensive	0.00%	0.00%	0.00%	0.18%	0.03%
	porn	2.86%	0.33%	0.00%	1.63%	0.08%
#langs =0% C		7	0	1	7	0
#langs <50% C		44	4	19	11	9
#langs >50% NL		13	0	0	7	1
#langs >50% WL		1	0	0	3	4

Averages of sentence-level annotations across datasets and selected languages



Fraction of languages in each dataset below a given quality threshold



(a) Monolingual corpora

(b) Parallel corpora

Percentage of sentences labeled as correct vs. log N sentences for all audited

languages

	es_XX	bm_ML	yo_NG	tr_TR	ku_TR	zh_CN	af_ZA	jv_ID	zh_TW	it_IT	mean
Acc-6	0.58	0.73	0.41	0.45	0.43	0.55	0.65	0.55	0.46	0.55	0.66
Acc-4	0.77	0.73	0.60	0.55	0.56	0.72	0.72	0.57	0.58	0.66	0.72
Acc-2	0.91	0.96	0.72	0.64	0.71	0.79	0.77	0.92	0.81	0.69	0.79

Rater evaluation for a subset of audits from CCAIined (translated from English) measured by the accuracy (Acc-n) of annotations by non-proficient speaker against annotations by proficient speakers

	tyv	rm	bar	eml	zh	la	mean
Acc-6	1.0	0.98	1.0	1.0	0.86	1.0	0.98
Acc-4	1.0	1.0	1.0	1.0	0.87	1.0	0.98
Acc-2	1.0	1.0	1.0	1.0	0.87	1.0	0.98

Rater evaluation for a subset of audits from OSCAR measured by the accuracy (Acc-n) of annotations by non-proficient speaker against annotations by proficient speakers

en	The prime minister of the UK is Boris Johnson .
nl	De minister-president van Nederland is Mark Rutte . en: The prime minister of the Netherlands is Mark Rutte.
en	24 March 2018
pt	14 Novembro 2018 en: 14 November 2018
en	The current local time in Sarasota is 89 minutes.
nn	Den lokale tiden i Miami er 86 minutt. en: The local time in Miami is 86 minutes.
en	In 1932 the highway was extended north to LA .
bar	1938 is de Autobahn bei Inglstod fertig gstellt. en: The highway near Inglstod was completed in 1938.

Examples of "parallel" data where the translation has a different meaning than the source, but the form looks the same. (We added translations of the non-English side.) Such data may encourage hallucinations of fake "facts"

VOCAB: (w/definition)

Automatic language classification: A process utilized in the article to automatically determine the language of web-derived datasets, aiding in the curation and analysis of multilingual data

Low-resource languages: Languages with limited available linguistic resources, which are a focus in the article due to their challenges in data curation and evaluation

Error taxonomy: A systematic classification system used to categorize and analyze different types of errors within datasets, as applied in the manual data audit

process

Web crawlers: Automated programs or algorithms designed to navigate the internet and collect data, as mentioned in the article in the context of obtaining web-derived collections for NLP research

LangID: A shorthand for Language Identification, referring to the process of automatically determining the language of a given piece of text, discussed in the article regarding its use in data curation

WMT benchmarks: Benchmarks from the Workshop on Machine Translation (WMT), which may serve as reference points or standards in the field of machine translation. The article does not explicitly mention WMT benchmarks, but it discusses benchmarks and challenges in the context of evaluating datasets

Macro-average: An averaging technique used to calculate overall performance metrics across different categories or languages, applied in the article to aggregate labels and assess dataset quality

Micro-average: A method of averaging that provides a per-instance assessment, employed in the article to offer a more detailed view of performance metrics on a sentence-by-sentence basis

Correct codes: Refers to data labeled accurately in terms of language codes, as discussed in the article concerning the assessment of correct sentences in the auditing process

Spearman rank correlation: A statistical measure used in the article to evaluate the correlation between different variables, such as the correlation between data quality scores and translation performance in downstream applications

<p>Cited references to follow up on</p>	<p>Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In <i>Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers</i>, pages 427–431, Valencia, Spain. Association for Computational Linguistics.</p> <p>Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. In <i>Proceedings of the Third Conference on Machine Translation: Shared Task Papers</i>, pages 888–895, Belgium, Brussels. Association for Computational Linguistics.</p>
<p>Follow up Questions</p>	<p>How can the proposed solutions for effective, low-effort data auditing and error taxonomy be practically implemented in the NLP community's current dataset curation practices?</p> <p>In the context of low-resource languages, what additional challenges and potential solutions should be considered to ensure the quality and inclusivity of multilingual datasets in future NLP research?</p> <p>Considering the identified issues with mislabeled language codes and low-quality content, what specific strategies and best practices can be recommended for the improvement of automated filtering methods to enhance the overall quality of multilingual datasets for NLP applications?</p>

Notes (written with the assistance of ChatGPT)

Introduction

- Improved access to multilingual datasets for NLP research
- Web-derived collections available for download: ParaCrawl, WikiMatrix, CCAligned, OSCAR, etc.
- Enabled development of highly multilingual models like mT5, M2M-100, M4
- Curating datasets relies on website clues and automatic language classification (LangID)
- Automatically crawled datasets tend to have lower overall quality than hand-curated collections
- Quality is judged through improvements in downstream applications
- Promising for low-resource languages, but there's a lack of research on evaluating data collections and crawling tools for such languages
- Manual data audit for 230 per-language subsets of five major crawled multilingual datasets
- Proposed solutions for effective, low-effort data auditing and an error taxonomy
- Quantitative analysis reveals surprisingly low amounts of valid in-language data and identifies systematic issues
- Many datasets labeled with nontransparent or incorrect language codes
- Reflection on potential harm of low-quality data releases for low-resource languages
- Recommendations provided for future multilingual data releases

Related Work

- Corpora collected by web crawlers are noisy
- Issues with web crawls of lower-resource languages, especially with segment-level LangID
- Cleaning and filtering web crawls can enhance general language modeling and downstream task performance
- Difficulty in validating automatically collected and curated datasets as ML research scales
- Focus on advancing methodologies and best practices for dataset validation
- Introduction of data statements by Bender and Friedman as a framework for dataset description
- Similar work in systematizing documentation in data science and machine learning
- Data quality implicitly documented by successes of filtering methods
- Literature on filtering data for various NLP tasks
- Analysis of a highly multilingual web crawl and LangID-related quality issues by Caswell et al.
- Brief analysis of the quality of OSCAR by Caswell et al., with focus on the presence of in-language content
- Dodge et al. automatically documented and analyzed the contents and sources of C4, revealing machine-translated contents and NLP benchmark data

Multilingual Corpora

- Table 1 overview of selected multilingual corpora
- Corpora selected for multilinguality and inclusion of under-studied languages in NLP
- Corpora derived from CommonCrawl (CC), except WikiMatrix and ParaCrawl

- CCAIined (El-Kishky et al., 2020): parallel dataset aligned with FastText LangID and cross-lingual LASER embeddings
- Multilingual C4 (mC4) (Xue et al., 2021): document-level dataset for mT5, filtered and deduplicated, language identification using CLD3
- OSCAR (Ortiz Suárez et al., 2019; Ortiz Suárez et al., 2020): monolingual corpora from CC snapshots, deduplicated, LangID at line-level
- ParaCrawl v7.1: parallel dataset with 41 language pairs, primarily aligned with English, mined using Bitextor
- WikiMatrix (Schwenk et al., 2021): public dataset with 135M parallel sentences in 1620 language pairs mined from Wikipedia
- Focus on language pairs with English on one side in the audit

Auditing Data Quality

Auditing Process

- Participants
 - 51 volunteers from NLP community, covering 70 languages
 - Each sentence annotated by one rater
 - Hypothesis tested with non-expert annotations
- Sample Selection
 - Random sample of 100 lines for each language in each dataset
 - Manual annotation based on error taxonomy
 - Focus on languages with the least sentences in each dataset
- Non-expert Labeling Strategies
 - Volunteers familiar with languages or used dictionaries/internet search
 - Emphasis on low-resource evaluation by non-proficient speakers
- Effort
 - Individual effort dependent on data quality and complexity
 - Effort varied based on annotator's knowledge of the language(s)
- Taxonomy
 - Error classes: Incorrect Translation (X), Wrong Language (WL), Non-Linguistic Content (NL)
 - Correct sentences (C) further classified into single words/phrases (CS) and boilerplate contents (CB)
 - Offensive/pornographic content flagged

Human Audit Results

- Interpretation of Results
 - Compute percentage of each label within 100 audited sentences for each language
 - Aggregate labels across languages: macro-average or weight them by dataset presence: micro-average
 - Combined statistics for correct codes (CC, CB, CS) as C

- Results based on a small sample partially annotated by non-experts
- Capture ratio of languages (25–55%), tiny fraction of overall sentences (0.00004–0.002%)
- Quality Issues Across Datasets
 - Macro-averaged results show varied ratio of correct samples (C) (24% to 87%)
 - Severe problems in CCAIghed and WikiMatrix, with many languages having under 50% correct sentences
 - Some datasets show a high percentage of misaligned/mistranslated sentences (X), especially in WikiMatrix
- Reporting Issues and Imbalance
 - Macro-average gives equal weight to low and high-resource languages
 - Micro-average (per-sentence basis) provides a more optimistic view
 - Evaluation and tuning often focused on higher-resource languages, leading to low-quality issues in underrepresented languages
- Nonlinguistic and Wrong Language Content
 - Nonlinguistic content more common than wrong-language content
 - CCAIghed has the highest percentage of nonlinguistic content (31.42%)
 - mC4 has the highest ratio of sentences in incorrect languages (15.98% average)
- Language Confusion
 - Languages confused often related to higher-resource languages
 - Some "out-of-model cousin" cases where unsupported languages end up in similar-seeming languages
- Quality and Size Correlation
 - Low-resource datasets tend to have lower human-judged quality
 - Positive correlation between quality (%C) and dataset size
- Languages with Lowest Quality
 - Poor quality for languages in romanized script (`_rom/_latn`) and African languages
 - Some languages have extremely low quality even within the same datasets
- Offensive and Pornographic Content
 - Overall, sampled sentences had low offensive content but notable amounts of pornographic content in CCAIghed for 11 languages
- Annotation Quality
 - Accuracy (Acc) of labels assigned by non-proficient speakers compared to proficient speakers measured
 - Mean accuracy of 0.66 for CCAIghed audits (6-class taxonomy) and 0.98 for OSCAR audits
 - Significant drop of accuracy for finer-grained labels suggests room for improvement in the error taxonomy

Automatic Filtering

- Given WL and NL frequency, tempting to use LangID for per-sentence filtering
- Sentence-level n-gram LangID filtering using CLD3 on CCAIghed

- Evaluation shows CLD3 average precision at 40.6%
- Sentence-level Transformer LangID filtering
 - Semi-supervised Transformer-based models outperform n-gram models
 - Transformer model applied to CCAIghed data
 - Filtering noisy corpora (<50% correct) boosts median precision from 13.8% to 43.9%
 - Cost: 77.5% recall loss
 - Biggest winners: Lingala (8% to 80%), Oromo (2% to 33% in-language)
 - Cost: Lose 50% of correct in-language sentences, reduced from 22k to 3k and 1k
 - Moral: No one-size-fits-all approach for sentence-level LangID filtering

Dataset Mislabeling

- Standardized Language Codes
 - Importance for practical data use and exchange
 - BCP-47 standard widely used, based on ISO639-2 and ISO639-3 codes
 - Allows subtags for scripts and regional varieties
 - Enhances transparency and interoperability, especially with growing language diversity in NLP
- Errors and Inconsistencies in Language Code Usage
 - Analysis includes JW300 dataset from jw.org
 - Find 8 nonstandard codes in CCAIghed, 3 in OSCAR, 1 in mC4, 1 in WikiMatrix, and 70 in JW300 (83 in total)
 - Excludes 59 codes affected by superset issues
- Inconsistent Language Codes
 - Using nonstandard or invented codes is a common issue
 - Examples: CCAIghed uses only two-letter codes, OSCAR mislabels Tosk Albanian as Allemannic
- False Sign Languages
 - 12% of JW300 carry sign language codes (48/417)
 - Instead of sign language transcripts, they contain texts in another high-resource language
 - Example: en-zsl (Zambian sign language) data is actually English-English parallel data
- Mysterious Supersets
 - Difficulty determining specific language when datasets contain supersets of other language codes
 - Example: WikiMatrix has Serbian (sr), Croatian (hr), Bosnian (bs), and Serbo-Croatian (sh) as a superset
- Deprecated Codes
 - Deprecated codes used in datasets: sh in WikiMatrix, iw in mC4, sh and eml in Oscar, daf in JW300

Risks of Low-Quality Data

- Low Quality in Downstream Applications

- Text corpora crucial for downstream NLP applications like question answering and text summarization
- Flawed data for original systems can lead to failures in derived technology for languages down the line
- Calls for future studies considering data size, domain, language-specific phenomena, and metric biases
- Impact on Translation Quality
 - Comparison of C% metric from audit with sentencepiece-BLEU (spBLEU) of M2M124 multilingual translation model
 - Positive correlation (Spearman) between data quality scores and spBLEU ($\rho = 0.44$, $p = 0.041$)
 - Correlation with data size is higher ($\rho = 0.66$, $p = 0.00078$)
 - Correlation between product of C% and data size is the highest ($\rho = 0.73$, $p = 0.00013$)
- Representation Washing
 - Datasets with many low-resource languages may create a false sense of progress and equity
 - Low-quality datasets as benchmarks may exaggerate model performance or incorrectly assume tasks are harder than they are
 - Risks redirecting effort away from tasks and languages that need attention
- Trust in Incorrect "Facts"
 - Instances of parallel-looking sentences structurally and semantically similar but not factually correct
 - Models may produce plausible but incorrect translations, leading to algorithmic trust
 - Automation bias amplifies issues of inaccurate translations caused by misalignments

Future Work and Recommendations

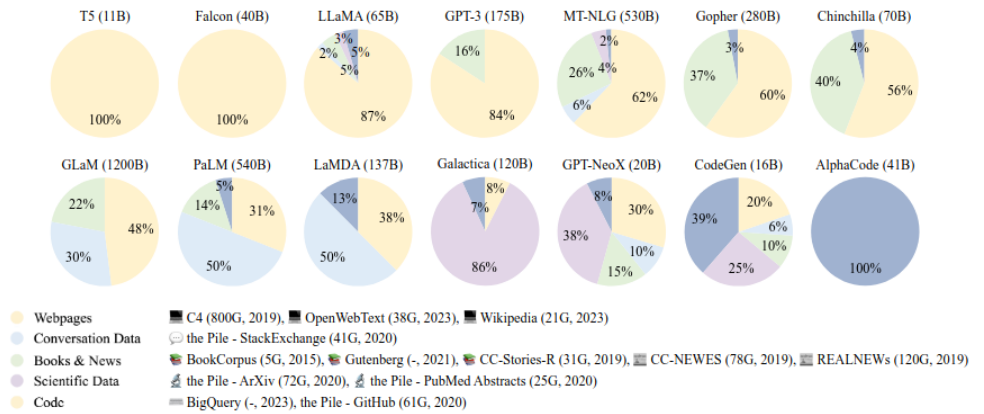
- Severe Quality Issues in Multilingual Corpora
 - Evaluation of five multilingual corpora reveals consistent and severe quality issues, especially in lower-resource languages
 - Samples of 205 languages rated, 87 with under 50% usable data, 15 at 0% in-language
 - Identified issues with mislabeled data and nonstandard language codes, particularly in JW300 dataset
 - 83 affected corpora identified, at least 48 entirely spurious
- Lack of Reported Quality Issues
 - Majority of quality issues had not been reported or investigated in depth
 - Issues might go unnoticed for languages not represented in crawling methods evaluation, causing harm in downstream applications
- Recommendations for Improvement
 - Improve ease and accuracy of human evaluation
 - Consider close dialects and develop a standard suite of automatic metrics for datasets
 - Address the estimated portion of a dataset generated by machine translation, language models, or bots/templates

- Strongly recommend looking at samples of any dataset before use or release
- Explore techniques for data quality improvement, such as length-ratio filtering, LangID, TF-IDF wordlists, dictionaries, and neural approaches like LM scoring
- Documentation as an alternative to filtering, with per-language quality scores and notes about known issues
- Importance of Continuing Evaluations
 - Encourage the community to continue conducting evaluations and audits of public datasets, similar to system comparison papers

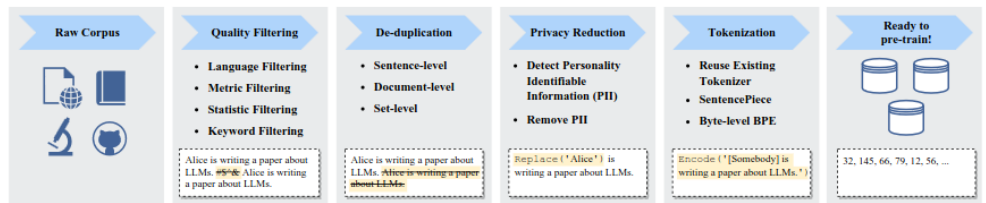
Article #14 Notes: A Survey of Large Language Models

Source Title	A Survey of Large Language Models
Source citation (APA Format)	<p>Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ... Wen, J.-R. (2023). <i>A survey of large language models</i>. (arXiv:2303.18223). arXiv. https://arxiv.org/abs/2303.18223</p>
Original URL	https://arxiv.org/abs/2303.18223
Source type	Journal Meeting
Keywords	Language Model Pre-training, Data Processing and Quality, Model Architectures, Decoding Strategies, Model Training Optimization
#Tags	#llms, #nlp, #dataquality, #llmpretraining, #trainingoptimizatoin
Summary of key points + notes (include methodology)	<p>The article delves into the pre-training phase of large language models (LLMs), emphasizing the critical elements of data collection, preprocessing, and the impact of quality pre-training data on model performance. It explores diverse data sources, including general and specialized text, and discusses challenges in filtering and processing, especially with web data. The study also investigates various model architectures, from traditional to emergent designs, shedding light on their advantages and efficiency. Decoding strategies post-pre-training are addressed, covering auto-regressive methods and strategies like greedy search and sampling. The article concludes with insights into model training optimization, detailing strategies for efficient batch training, learning rate, and scalable techniques such as 3D parallelism. Overall, the methodology involves a comprehensive examination of the key components in LLM pre-training, offering insights into both established practices and emerging trends in the field.</p>
Research Question/Problem/Need	What are the key challenges and optimization strategies in the pre-training phase of large language models, encompassing data collection, model architectures, decoding strategies, and efficient training methodologies?

Important Figures



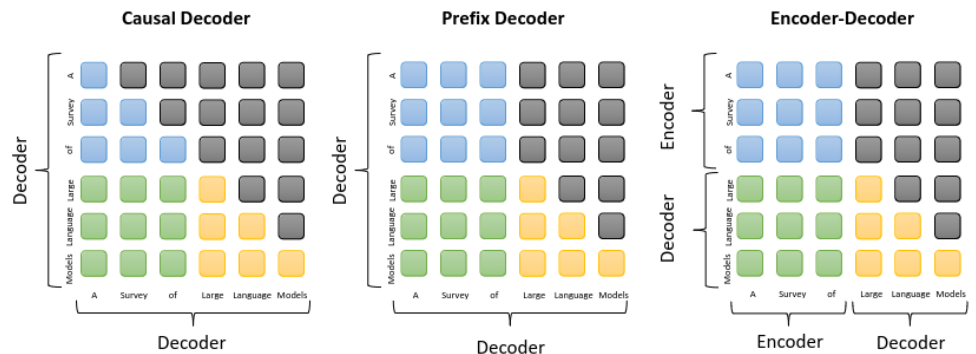
Ratios of various data sources in the pre-training data for existing LLMs



An illustration of a typical data preprocessing pipeline for pre-training large language models

Model	Category	Size	Normalization	PE	Activation	Bias	#L	#H	d_{model}	MCL
GPT3 [55]	Causal decoder	175B	Pre LayerNorm	Learned	GeLU	✓	96	96	12288	2048
PanGU- α [75]	Causal decoder	207B	Pre LayerNorm	Learned	GeLU	✓	64	128	16384	1024
OPT [81]	Causal decoder	175B	Pre LayerNorm	Learned	ReLU	✓	96	96	12288	2048
PaLM [56]	Causal decoder	540B	Pre LayerNorm	RoPE	SwiGLU	×	118	48	18432	2048
BLOOM [69]	Causal decoder	176B	Pre LayerNorm	ALiBi	GeLU	×	70	112	14336	2048
MT-NLG [100]	Causal decoder	530B	-	-	-	-	105	128	20480	2048
Gopher [59]	Causal decoder	280B	Pre RMSNorm	Relative	-	-	80	128	16384	2048
Chinchilla [34]	Causal decoder	70B	Pre RMSNorm	Relative	-	-	80	64	8192	-
Galactica [35]	Causal decoder	120B	Pre LayerNorm	Learned	GeLU	×	96	80	10240	2048
LaMDA [63]	Causal decoder	137B	-	Relative	GeLU	×	64	128	8192	-
Jurassic-1 [94]	Causal decoder	178B	Pre LayerNorm	Learned	GeLU	✓	76	96	13824	2048
LLaMA [57]	Causal decoder	65B	Pre RMSNorm	RoPE	SwiGLU	×	80	64	8192	2048
LLaMA 2 [90]	Causal decoder	70B	Pre RMSNorm	RePE	SwiGLU	×	80	64	8192	4096
Falcon [127]	Causal decoder	40B	Pre LayerNorm	RoPE	GeLU	×	60	64	8192	2048
GLM-130B [84]	Prefix decoder	130B	Post DeepNorm	RoPE	GeLU	✓	70	96	12288	2048
T5 [73]	Encoder-decoder	11B	Pre RMSNorm	Relative	ReLU	×	24	128	1024	512

Model cards of several selected LLMs with public configuration details



A comparison of the attention patterns in three mainstream architectures. Here, the blue, green, yellow and grey rounded rectangles indicate the attention between prefix tokens, attention between prefix and target tokens, attention between target tokens, and masked attention respectively

Configuration	Method	Equation
Normalization position	Post Norm [22]	$\text{Norm}(\mathbf{x} + \text{Sublayer}(\mathbf{x}))$
	Pre Norm [26]	$\mathbf{x} + \text{Sublayer}(\text{Norm}(\mathbf{x}))$
	Sandwich Norm [201]	$\mathbf{x} + \text{Norm}(\text{Sublayer}(\text{Norm}(\mathbf{x})))$
Normalization method	LayerNorm [202]	$\frac{\mathbf{x} - \mu}{\sqrt{\sigma}} \cdot \gamma + \beta, \quad \mu = \frac{1}{d} \sum_{i=1}^d x_i, \quad \sigma = \sqrt{\frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2}$
	RMSNorm [203]	$\frac{\mathbf{x}}{\text{RMS}(\mathbf{x})} \cdot \gamma, \quad \text{RMS}(\mathbf{x}) = \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2}$
	DeepNorm [204]	$\text{LayerNorm}(\alpha \cdot \mathbf{x} + \text{Sublayer}(\mathbf{x}))$
Activation function	ReLU [205]	$\text{ReLU}(\mathbf{x}) = \max(\mathbf{x}, 0)$
	GeLU [206]	$\text{GeLU}(\mathbf{x}) = 0.5\mathbf{x} \otimes [1 + \text{erf}(\mathbf{x}/\sqrt{2})], \quad \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$
	Swish [207]	$\text{Swish}(\mathbf{x}) = \mathbf{x} \otimes \text{sigmoid}(\mathbf{x})$
	SwiGLU [208]	$\text{SwiGLU}(\mathbf{x}_1, \mathbf{x}_2) = \text{Swish}(\mathbf{x}_1) \otimes \mathbf{x}_2$
	GeGLU [208]	$\text{GeGLU}(\mathbf{x}_1, \mathbf{x}_2) = \text{GeLU}(\mathbf{x}_1) \otimes \mathbf{x}_2$
Position embedding	Absolute [22]	$\mathbf{x}_i = \mathbf{x}_i + \mathbf{p}_i$
	Relative [23]	$A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{x}_j^T \mathbf{W}_k^T + r_{i-j}$
	RoPE [209]	$A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{R}_{\theta, i-j} \mathbf{x}_j^T \mathbf{W}_k^T$
	Alibi [210]	$A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{R}_{\theta, i-j} \mathbf{x}_j^T \mathbf{W}_k^T, A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{x}_j^T \mathbf{W}_k^T - m(i-j)$

Detailed formulations for the network configurations

I am sleepy. I start a pot of _____

coffee	0.661	strong	0.008	soup	0.005
water	0.119	black	0.008
tea	0.057	hot	0.007	happy	4.3e-6
rice	0.017	oat	0.006	Boh	4.3e-6
chai	0.012	beans	0.006

The probability distribution over the vocabulary in descending order for the next token of the context "I am sleepy. I start a pot of"

Model	Batch Size (#tokens)	Learning Rate	Warmup	Decay Method	Optimizer	Precision Type	Weight Decay	Grad Clip	Dropout
GPT3 (175B)	32K→3.2M	6×10^{-5}	yes	cosine decay to 10%	Adam	FP16	0.1	1.0	-
PanGu- α (200B)	-	2×10^{-5}	-	-	Adam	-	0.1	-	-
OPT (175B)	2M	1.2×10^{-4}	yes	manual decay	AdamW	FP16	0.1	-	0.1
PaLM (540B)	1M→4M	1×10^{-2}	no	inverse square root	Adafactor	BF16	l_r^2	1.0	0.1
BLOOM (176B)	4M	6×10^{-5}	yes	cosine decay to 10%	Adam	BF16	0.1	1.0	0.0
MT-NLG (530B)	64 K→3.75M	5×10^{-5}	yes	cosine decay to 10%	Adam	BF16	0.1	1.0	-
Gopher (280B)	3M→6M	4×10^{-5}	yes	cosine decay to 10%	Adam	BF16	-	1.0	-
Chinchilla (70B)	1.5M→3M	1×10^{-4}	yes	cosine decay to 10%	AdamW	BF16	-	-	-
Galactica (120B)	2M	7×10^{-6}	yes	linear decay to 10%	AdamW	-	0.1	1.0	0.1
LaMDA (137B)	256K	-	-	-	-	BF16	-	-	-
Jurassic-1 (178B)	32 K→3.2M	6×10^{-5}	yes	-	-	-	-	-	-
LLaMA (65B)	4M	1.5×10^{-4}	yes	cosine decay to 10%	AdamW	-	0.1	1.0	-
LLaMA 2 (70B)	4M	1.5×10^{-4}	yes	cosine decay to 10%	AdamW	-	0.1	1.0	-
Falcon (40B)	2M	1.85×10^{-4}	yes	cosine decay to 10%	AdamW	BF16	0.1	-	-
GLM (130B)	0.4M→8.25M	8×10^{-5}	yes	cosine decay to 10%	AdamW	FP16	0.1	1.0	0.1
T5 (11B)	64K	1×10^{-2}	no	inverse square root	AdaFactor	-	-	-	0.1
ERNIE 3.0 Titan (260B)	-	1×10^{-4}	-	-	Adam	FP16	0.1	1.0	-
PanGu- Σ (1.085T)	0.5M	2×10^{-5}	yes	-	Adam	FP16	-	-	-

Detailed optimization settings of several existing LLMs

VOCAB: (w/definition)

Tokenization: In the context of the article, tokenization refers to the crucial step in data preprocessing where raw text is segmented into individual tokens for input into Large Language Models (LLMs). The article mentions various tokenization methods, including Byte-Pair Encoding (BPE), WordPiece, and Unigram, with a special emphasis on customized tokenizers using the SentencePiece library for diverse corpora

SentencePiece: SentencePiece is highlighted as a library for customized tokenization in the article. It is mentioned in the context of creating tailored

tokenizers, which proves beneficial for handling diverse corpora during the tokenization process.

Double descent: The term "double descent" is not explicitly mentioned in the provided article notes. However, in machine learning, double descent typically refers to a phenomenon where, as model complexity increases, the test error first decreases, then increases, and finally decreases again. This concept may be indirectly related to discussions in the article about the quality of pre-training data and its impact on the performance of Large Language Models

In-context learning: In-context learning is associated with the capability of certain architectures, such as the Causal Decoder Architecture, to perform autoregressive generation by attending only to past tokens and themselves. GPT-3, developed on this architecture, is highlighted for its superior in-context learning capability

Unidirectional attention mask: The unidirectional attention mask is a component of the Causal Decoder Architecture mentioned in the article. It limits the attention of input tokens to past tokens and themselves, enabling autoregressive generation and enhancing the model's in-context learning capability

MoE scaling: MoE Scaling refers to the application of a mixture-of-experts approach in the scaling of neural network weights. This technique involves sparsely activating a subset of weights for each input. The article mentions that models like Switch Transformer and GLaM implement MoE scaling, leading to substantial performance improvement with an increased number of experts or total parameter size.

Parameterized state space models: Parameterized state space models are part of the emergent architectures discussed in the article. These models include S4, GSS, and H3, and they aim to address the quadratic computational complexity issues associated with conventional Transformer architectures, allowing for more efficient training and inference with long inputs.

Recursive update mechanisms: Recursive update mechanisms are mentioned in the context of emergent architectures in the article. Architectures like RWKV, RetNet, and others employ recursive update mechanisms, combining features of both recurrent neural networks (RNNs) and Transformer-like architectures. This enables highly parallel and efficient training with GPU parallelism techniques, contributing to overall model efficiency.

Cited references to follow up on	<p>[112] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. P. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. S. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui, "Glam: Efficient scaling of language models with mixture-of-experts," in <i>International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, 2022</i>, pp. 5547–5569.</p>
Follow up Questions	<p>How do different decoding strategies balance output diversity and coherence, and are there specific scenarios where one strategy is more effective?</p> <p>In emerging architectures like parameterized state space models, how do improvements in computational efficiency impact training and inference, and how do these architectures address challenges in conventional Transformer models with longer input sequences?</p> <p>Considering the importance of data quality, how do proposed data preprocessing techniques like privacy reduction and deduplication impact model performance, and what potential biases may arise in classifier-based approaches, especially in dialectal languages?</p>

Notes (on relevant sections with the assistance of ChatGPT)

Pre-training

- Pre-training establishes essential language understanding and generation skills for LLMs
- Large-scale corpora critical for scale and quality of pre-training
- Model architectures, acceleration methods, and optimization techniques crucial for effective pre-training

Data Collection

Data Source

- Pre-training Corpus Sources
 - Two broad categories: general data and specialized data
- General Text Data
 - *Webpages*: Diverse but requires careful filtering
 - *Conversation Text*: Enhances conversational competence; risk of misinterpretation
 - *Books*: Formal, long texts beneficial for learning linguistic knowledge
- Specialized Text Data
 - *Multilingual Text*: Improves multilingual abilities; models like BLOOM and PaLM demonstrate strong performance
 - *Scientific Text*: Enhances understanding of scientific knowledge; relies on arXiv papers, scientific textbooks, and math webpages
 - *Code*: Significant for program synthesis; two common sources are programming Q&A communities (e.g., Stack Exchange) and public repositories (e.g., GitHub)
- Challenges
 - Filtering and processing are crucial for ensuring data quality, especially in web data
 - Risk of integrating dialogue data may lead to a decline in efficacy
 - Specialized datasets require specific tokenization and preprocessing techniques

Data Preprocessing

- Data Preprocessing
 - Essential after collecting large text data
 - Aim: Construct a high-quality pre-training corpus
- Quality Filtering
 - Two approaches: classifier-based and heuristic-based
 - Language, metric, statistic, and keyword-based filtering methods
 - Potential bias in classifier-based approaches, especially in dialectal languages
- Deduplication
 - Reduces corpus diversity issues
 - Perform at different granularities: sentence, document, and dataset-level
 - Essential to prevent dataset contamination

- Privacy Reduction
 - Remove personally identifiable information (PII) to mitigate privacy risks
 - Rule-based methods (e.g., keyword spotting) used for PII detection
- Tokenization
 - Crucial step for data preprocessing
 - Aims to segment raw text into individual tokens for LLM inputs
 - Different methods: Byte-Pair Encoding (BPE), WordPiece, and Unigram
 - Customized tokenizers with SentencePiece library are beneficial for diverse corpora

Effect of Pre-training Data on LLMs

- Iterative Pre-training
 - Usually infeasible for LLMs due to high computational demand
 - Emphasizes the need for a well-prepared pre-training corpus
- Mixture of Sources
 - Diverse sources provide distinct linguistic characteristics and semantic knowledge
 - Mixing sources enhances LLM's generalization capacity
 - Careful consideration of data distribution crucial for downstream task performance
 - Excessive data from one domain can impact generalization to other domains
- Amount of Pre-training Data
 - Sufficient high-quality data essential for effective LLM pre-training
 - Model size and data size scaling are interrelated for optimal performance
 - Neglecting data quantity can lead to sub-optimal training, affecting performance
- Quality of Pre-training Data
 - Low-quality corpus (noisy, toxic, duplicate) harms LLM performance
 - Quantity and quality of training data both crucial
 - Filtering and cleaning data improves LLM performance, prevents issues like "double descent"
 - Duplicate data degrades LLMs' ability to copy from context, affecting generalization
 - Preprocessing methods essential for stability in training process and overall model performance

Architectures

Typical Architectures

- Encoder-decoder Architecture
 - Vanilla Transformer model based on encoder-decoder architecture
 - Encoder uses multi-head self-attention layers to encode input
 - Decoder performs cross-attention on representations, autoregressively generates target sequence
 - Effective for NLP tasks (e.g., T5 and BART)
 - Few LLMs built on this architecture (e.g., Flan-T5)
- Causal Decoder Architecture

- Incorporates unidirectional attention mask in decoder
- Input tokens attend only to past tokens and themselves
- GPT-series models (e.g., GPT-3) developed on this architecture
- GPT-3 shows superior in-context learning capability
- Scaling plays a crucial role in increasing model capacity
- Adopted widely by various LLMs (e.g., OPT, BLOOM, Gopher)
- Prefix Decoder Architecture
 - Also known as non-causal decoder
 - Revises masking mechanism to enable bidirectional attention over prefix tokens
 - Unidirectional attention on generated tokens
 - Shared parameters during encoding and decoding
 - Practical suggestion: Continually train causal decoders and then convert them into prefix decoders
 - Representative LLMs based on prefix decoders include GLM-130B and U-PaLM
- Mixture-of-Experts (MoE) Scaling
 - Subset of neural network weights sparsely activated for each input
 - Implemented in models like Switch Transformer and GLaM
 - Substantial performance improvement with increased number of experts or total parameter size
- Emergent Architectures
 - Conventional Transformer architectures suffer from quadratic computational complexity
 - Efficiency becomes crucial for training and inference with long inputs
 - New architectures include parameterized state space models (e.g., S4, GSS, H3), long convolutions (e.g., Hyena), and Transformer-like architectures with recursive update mechanisms (e.g., RWKV, RetNet)
 - Key merits: Recursive output generation like RNNs and parallel encoding of entire sentences like Transformers
 - Enables highly parallel and efficient training with GPU parallelism techniques

Detailed Configuration

- Normalization Methods
 - *LayerNorm*: Original normalization method in vanilla Transformer, calculates mean and variance per layer
 - *RMSNorm*: Introduced for faster training, scales activations with root mean square (RMS), used in models like Gopher and Chinchilla
 - *DeepNorm*: Microsoft's method for stabilizing training in deep Transformers, adopted by GLM-130B
- Normalization Position
 - *Post-LN*: Placed between residual blocks, tends to be unstable and rarely used alone
 - *Pre-LN*: Applied before each sub-layer, more stable than post-LN, commonly adopted despite slightly decreased performance

- *Sandwich-LN*: Adds extra LN before residual connections, aimed at preventing value explosion, but may lead to training collapse
- Activation Functions
 - *GeLU Activations*: Widely used in existing LLMs for feed-forward networks
 - *SwiGLU and GeGLU Variants*: Variations of GLU activation used in recent LLMs like PaLM and LaMDA, achieving better performance but with additional parameters
- Position Embeddings
 - *Absolute Position Embedding*: Employed in vanilla Transformer, includes sinusoidal and learned position embeddings
 - *Relative Position Embedding*: Generated based on offsets between keys and queries, facilitates extrapolation to longer sequences
 - *Rotary Position Embedding (RoPE)*: Uses rotatory matrices based on absolute position, widely adopted in recent LLMs like PaLM and LLaMA, improving translation invariance and length extrapolation
 - *ALiBi*: Improves extrapolation by biasing attention scores with penalties based on distances, stable in BLOOM
- Attention Mechanism
 - *Full Attention*: Original pairwise attention mechanism in vanilla Transformer, quadratic complexity for all token pairs
 - *Sparse Attention*: Efficient variants like Factorized Attention adopted in GPT-3, allows each query to attend to a subset of tokens
 - *Multi-query/Grouped-query Attention*: Shares linear transformation matrices, reducing computation costs; explored in models like PaLM and StarCoder
 - *FlashAttention*: Optimizes speed and memory consumption on GPUs, utilizing different levels of memory for improved efficiency
 - *PagedAttention*: Addresses GPU memory occupation issues by partitioning sequences into subsequences, improving memory efficiency and throughput

Pre-training Tasks

- Language Modeling (LM)
 - Commonly used pre-training task for decoder-only LLMs like GPT-3 and PaLM
 - Objective is to autoregressively predict target tokens based on preceding tokens in a sequence
 - Training objective maximizes the likelihood: $LLM(x) = \sum_i \log P(x_i | x_{<i})$ for a sequence $x = \{x_1, \dots, x_n\}$
 - Decoder-only LLMs naturally transfer to certain tasks without fine-tuning, revealing task-universal capabilities
 - Variant: Prefix Language Modeling
 - Designed for pre-training models with the prefix decoder architecture
 - Randomly selected prefix tokens excluded in computing the loss
 - Performs slightly worse than traditional language modeling due to fewer tokens involved in pre-training

- Denoising Autoencoding (DAE)
 - Another widely used pre-training task, adopted by models like T5 and GLM-130B
 - Inputs $x \sim x$ for DAE are corrupted text with randomly replaced spans
 - Language models trained to recover replaced tokens \tilde{x}
 - Training objective: $L_{DAE}(x) = \log P(\tilde{x} | x \setminus \tilde{x})$
 - Implementation of DAE task is more complex than LM task, limiting its widespread usage for large language models
- Mixture-of-Denoisers (MoD)
 - Unified objective introduced for pre-training language models, combining LM and DAE objectives
 - Recognizes LM and DAE as different denoising tasks: S-denoiser (LM), R-denoiser (DAE, short span and low corruption), and X-denoiser (DAE, long span or high corruption)
 - For sentences starting with different special tokens ($\{[R], [S], [X]\}$), corresponding denoisers are optimized
 - Applied in the latest PaLM 2 model

Decoding Strategy

- Background
 - Post-pre-training, choosing an effective decoding strategy is crucial for generating appropriate outputs from LLMs
- Auto-Regressive Decoding
 - For decoder-only LLMs pre-trained on language modeling tasks
 - Greedy Search
 - Predicts the most likely token at each step based on previously generated tokens
 - Modeled as: $x_i = \arg \max P(x | x_{<i})$ for the token at the i -th step
 - Suitable for tasks where output depends heavily on input
 - In open-ended tasks, may lead to awkward and repetitive sentences
 - Sampling-Based Methods
 - Randomly selects the next token based on the probability distribution to enhance randomness
 - Modeled as: $x_i \sim P(x | x_{<i})$
 - Enhances diversity during generation
- Decoding Improvements for Greedy Search
 - Beam Search
 - Retains sentences with the n (beam size) highest probabilities at each step and selects the top probability response
 - Beam size configured within the range of 3 to 6
 - Length Penalty
 - Mitigates favoring shorter sentences in beam search
 - Normalizes sentence probability based on length (divided by an exponential power α)

- Decoding Improvements for Random Sampling
 - Temperature Sampling
 - Modulates randomness of sampling by adjusting the temperature coefficient of the softmax function
 - Reducing temperature increases the chance of selecting high-probability words
 - Top-k Sampling
 - Truncates tokens with lower probability and samples only from tokens with the top k highest probabilities
 - Top-p Sampling
 - Gradually adds tokens from the vocabulary sorted by generative probability until cumulative value exceeds p
- Efficient Decoding Strategies
 - Speculative Decoding
 - Uses a compact, efficient model (e.g., n-gram model) to generate short segments, verified and corrected by the LLM
 - Achieves notable speedup without compromising quality
 - Token-Level Early-Exit Techniques
 - Enables generating a token at lower Transformer layers for greater speedup but at the cost of quality
- Practical Settings
 - Various decoding strategies supported by existing libraries and public APIs
 - T5 [73]: Greedy search (default) and beam search (beam size 4) with length penalty 0.6 for translation and summarization
 - GPT-3 [55]: Beam search (beam size 4) with length penalty 0.6 for all tasks
 - Alpaca [128]: Sampling-based strategies with top-k (k = 50), top-p (p = 0.9), and temperature 0.7 for open-ended generation
 - LLaMA [57]: Greedy search for question answering, sampling with temperature settings 0.1 (pass@1) and 0.8 (pass@100) for code generation
 - OpenAI API: Supports greedy search, beam search, temperature sampling, and nucleus sampling, with additional parameters to control repetition degree

Model Training

- Optimization Setting
 - Commonly used settings for batch training, learning rate, optimizer, and training stability
- Batch Training
 - Language model pre-training often uses a large batch size (e.g., 2,048 examples or 4M tokens) to enhance stability and throughput
 - Dynamic batch size increase during training, demonstrated in GPT-3, improves training stability
- Learning Rate
 - Linear warm-up schedule (0.1% to 0.5% of steps) increases learning rate to maximum (e.g., 5×10^{-5} to 1×10^{-4})

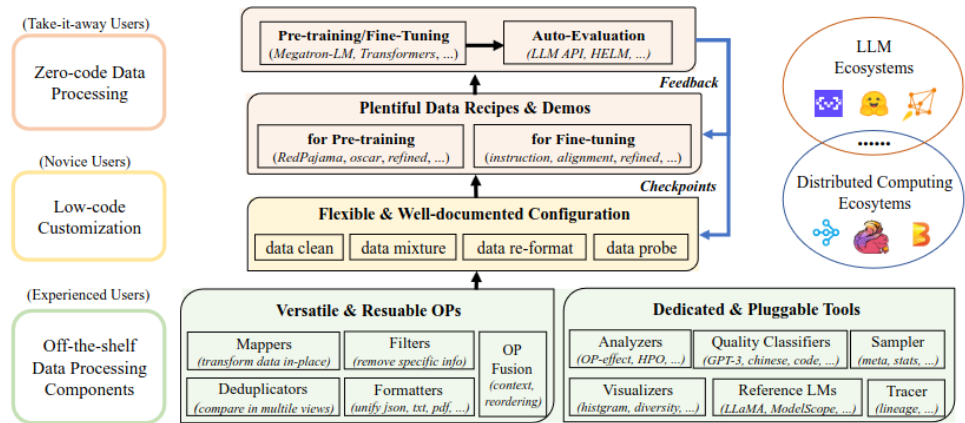
- Cosine decay strategy follows, reducing learning rate to 10% of maximum until convergence
- **Optimizer**
 - Adam optimizer and AdamW optimizer widely used (e.g., GPT-3)
 - Hyper-parameters: $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$
 - Adafactor optimizer (used in PaLM and T5) designed to conserve GPU memory
 - Hyper-parameters: $\beta_1 = 0.9$, $\beta_2 = 1.0 - k^{-0.8}$
- **Stabilizing the Training**
 - Weight decay and gradient clipping commonly used to address training instability
 - Threshold of gradient clipping: 1.0, weight decay rate: 0.1
 - PaLM and OPT use a strategy to restart training from an earlier checkpoint before spikes in training loss
 - GLM proposes shrinking the embedding layer gradients to alleviate abnormal gradients
- **Scalable Training Techniques**
 - Challenges with increasing model and data sizes necessitate efficient training strategies
 - Review of approaches: 3D parallelism, ZeRO, mixed precision training
- **3D Parallelism**
 - Combination of data parallelism, pipeline parallelism, and tensor parallelism
 - Data parallelism: Replicates model parameters across GPUs for scalable training
 - Pipeline parallelism: Distributes layers into multiple GPUs, using techniques to reduce inefficiencies
 - Tensor parallelism: Decomposes LLMs for multi-GPU loading, supported in open-source libraries
- **ZeRO**
 - DeepSpeed's technique addresses memory redundancy in data parallelism
 - Retains only a fraction of data on each GPU, reducing memory usage without increasing communication overhead
 - Three solutions: optimizer state partitioning, gradient partitioning, and parameter partitioning
- **Mixed Precision Training**
 - Transition from 32-bit FP32 to 16-bit floating-point numbers (FP16) for reduced memory usage and communication overhead
 - Some studies explore Brain Floating Point (BF16) for improved representation accuracy during pre-training
- **Overall Training Suggestion**
 - Joint use of training techniques, including 3D parallelism, ZeRO, and mixed precision training, to enhance throughput and model loading
 - Open-source libraries (e.g., DeepSpeed, Colossal-AI, Alpa) support parallel training methods
 - Techniques like ZeRO, FSDP, and activation recomputation reduce memory redundancy
 - Early detection of issues and performance prediction facilitated by mechanisms like GPT-4's predictable scaling

- Leveraging supporting training techniques in mainstream deep learning frameworks (e.g., PyTorch's FSDP) can enhance efficiency

Article #15 Notes: Data-Juicer: A One-Stop Data Processing System for Large Language Models

Source Title	Data-Juicer: A One-Stop Data Processing System for Large Language Models
Source citation (APA Format)	Chen, D., Huang, Y., Ma, Z., Chen, H., Pan, X., Ge, C., Gao, D., Xie, Y., Liu, Z., Gao, J., Li, Y., Ding, B., & Zhou, J. (2023). <i>Data-Juicer: A one-stop data processing system for large language models</i> . (arXiv:2309.02033). arXiv. https://arxiv.org/abs/2309.02033
Original URL	https://arxiv.org/abs/2309.02033
Source type	Journal Article
Keywords	Large Language Models (LLMs), Data-Juicer, Language Model Data Processing, Operator Pool for Data Processing, Feedback-Driven Data Processing
#Tags	#llms, #nlp, #pretraining, #distributeddataprocessing, #systemoptimization
Summary of key points + notes (include methodology)	The article highlights the significance of Large Language Models (LLMs) in achieving unprecedented intelligence, emphasizing their potential for artificial general intelligence. It identifies challenges in LLM data processing, such as heterogeneity, timely feedback, usability, customizability, and massive data volume. The proposed solution, Data-Juicer, is introduced as a comprehensive data processing system that addresses these challenges. The system features a Standardized Operator Pool, Feedback-Driven Data Processing with a dynamic feedback loop, Boosting Usability with Built-Ins, and Comprehensive System Optimization. Data-Juicer's methodology involves a unified configuration paradigm, a range of dedicated tools, and operator reordering for optimization. The article concludes with a quantitative evaluation demonstrating the system's superiority over a state-of-the-art baseline in terms of processing time, memory utilization, and CPU efficiency, showcasing its effectiveness in enhancing data quality and system scalability for LLMs.
Research Question/Problem/Need	How can Data-Juicer address the challenges in Large Language Model data processing, offering a unified and optimized solution to enhance usability, data quality, and scalability in the pursuit of artificial general intelligence?

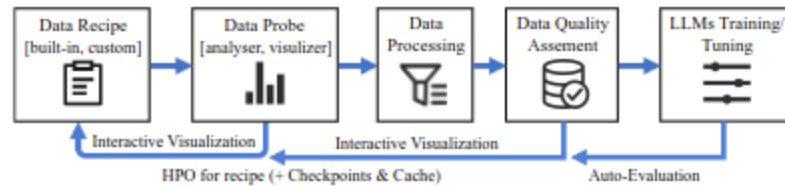
Important Figures



Overview of Data-Juicer

Category	Function	Input	Process Level	Output	OP Usage Examples
Formatters	Data format unifying	Dataset	Dataset	Dataset	Load and unify dataset-hub, txt, json, md, codes, html, pdf, docx, ...
Mappers	In-place text editing	Sample	Single-sample; Multi-samples	Sample; Samples	Transform specified headers, textual elements; Fix messy codes; Enable text enhancement
Filters	Conditional text removing	Sample	Single-sample; Dataset	Boolean	Filter by meta-info, stats (e.g., lines count); model scores; external resources (e.g., flagged words)
Deduplicators	Duplication removing	Single or Paired Dataset	Dataset	Dataset	Compare with hash-based and vector-based deduplication methods

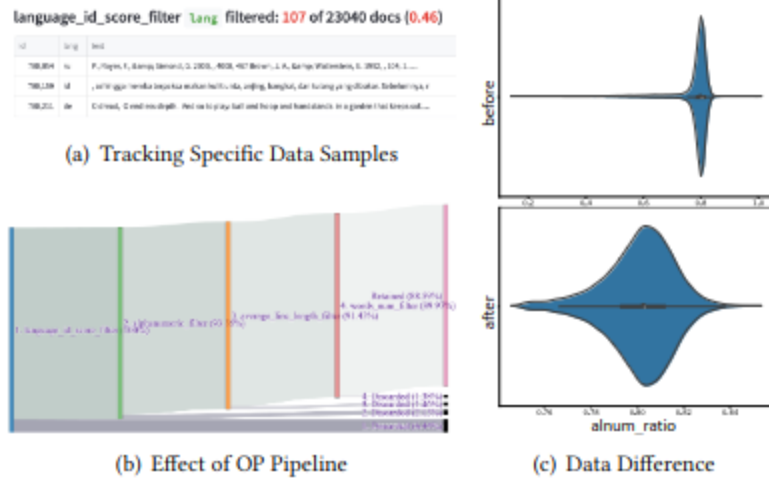
Overview of the operator (OP) pool in Data-Juicer, with a detailed list continuously maintained at the official documentation:



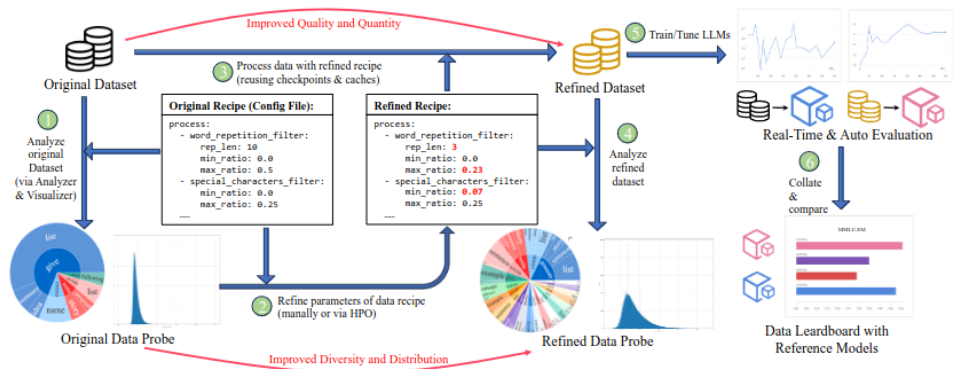
The feedback loop of Data-Juicer



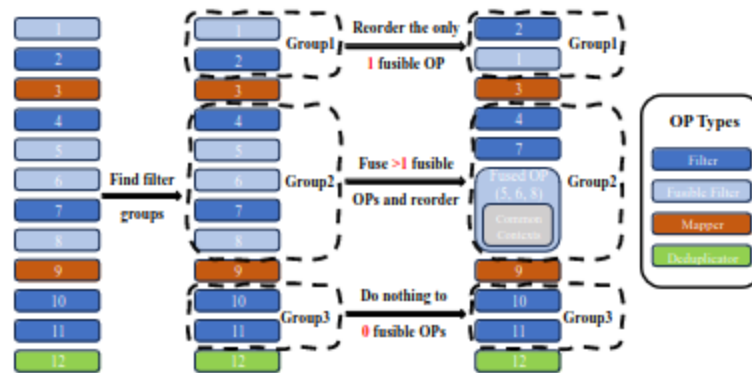
The demonstration of hyper-parameter optimization for data processing



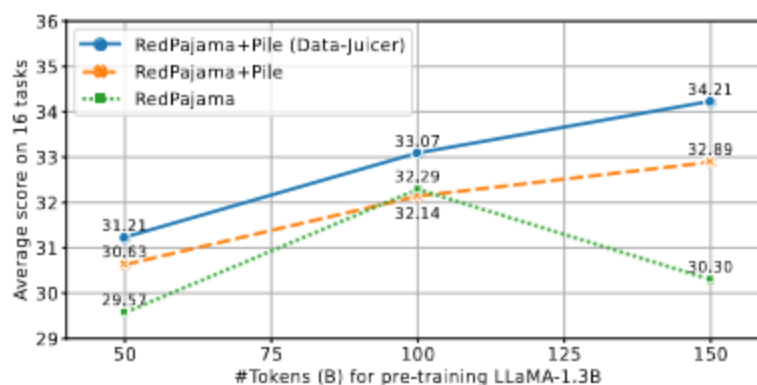
The illustration of several interactive visualization features of Data-Juicer



The demonstration of data processing feedback of Data-Juicer, which helps to generate better data recipes for LLMs



The OP fusion procedure for an OP list



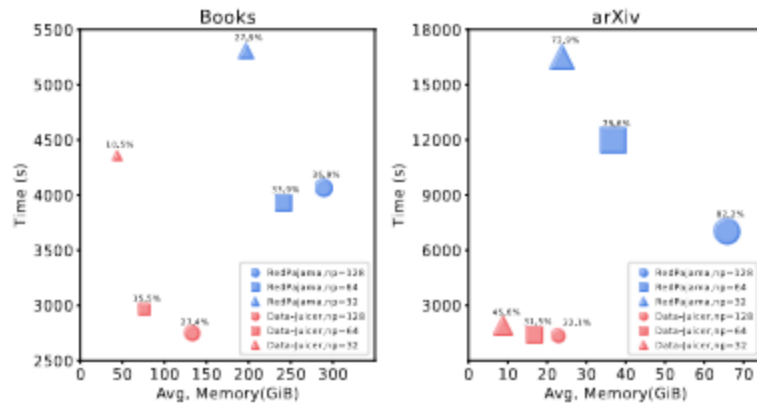
Evaluation results of reference models trained with different datasets but the same pre-training procedures

Model	Training Data	#Tokens	Score
Falcon-1.3B [40]	RefinedWeb	350B	33.97
Pythia-1.4B [28]	Pile	300B	33.96
LLaMA-1.3B	Data-Juicer (RedPajama+Pile)	150B	34.21
	+ Alpaca-CoT-IFT	150B + 15B	35.04
	+ Our Refined IFT	150B + 4.7B	36.76

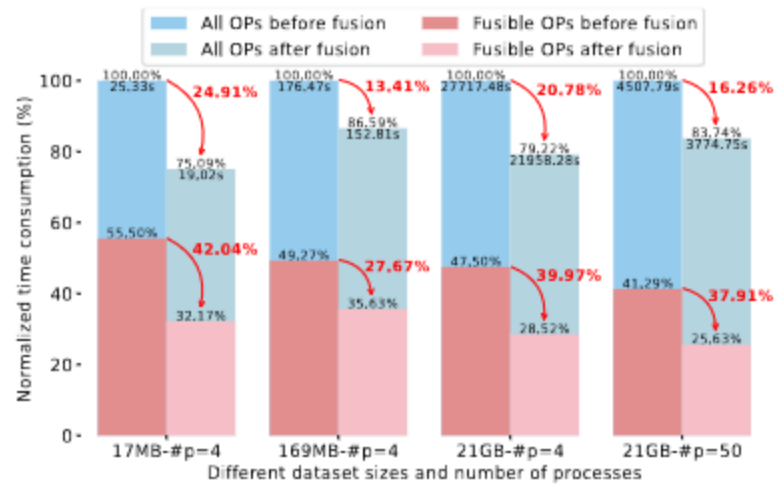
The average score of the pre-trained LLMs on 16 HELM core tasks

Model	Tuning Data	#Samples	Win	Tie
LLaMA-7B [33]	Alpaca	52k	16	100
	Data-Juicer	40k	44	
	Random (CFT, EN)	40k	19	105
	Data-Juicer	40k	36	
LLaMA2-7B (Chinese, FlagAlpha [41])	Belle	543k	28	99
	Data-Juicer	52k	33	
	Random (CFT, ZH)	52k	19	96
	Data-Juicer	52k	45	

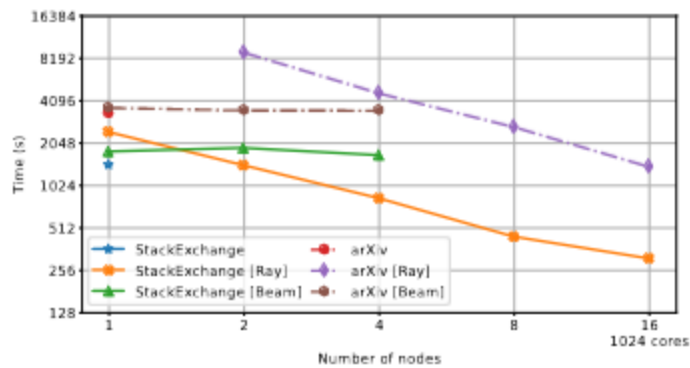
Results of pair-wise model comparisons using GPT4 scoring



Comparison of stand-alone processing performance, the marker size is proportional to average CPU usage (↓)



Time comparison before and after OP fusion



Scalability of Data-Juicer

VOCAB: (w/definition)

Heterogeneity: The presence of diverse and varied elements within LLM data, requiring specialized processing methods to handle different formats and sources.

	<p>Data-Juicer: A proposed data processing system designed to address challenges in LLM data processing, providing fine-grained abstraction, timely feedback, and end-to-end configurability.</p> <p>Operator Pool (OP): A collection of standardized operations in Data-Juicer, categorized as Formatters, Mappers, Filters, and Deduplicators, offering flexibility and user interaction in processing LLM data.</p> <p>Hyper-Parameter Optimization (HPO): The tuning of parameters in Data-Juicer for optimal data processing, tied to custom target metrics and visualization results for effective optimization.</p> <p>Checkpoint and Caching: Built-in mechanisms in Data-Juicer for resilient and reliable data processing, allowing swift recovery during system restarts, mitigating redundancy, and providing flexibility in space-time trade-offs.</p> <p>Interactive Visualization: A feature in Data-Juicer integral to feedback stages, enabling users to visually track the effects of individual operations, compare results before and after processing, and enhance control over data processing.</p> <p>Distributed Data Processing: The capability of Data-Juicer to work seamlessly with distributed processing frameworks like Ray, Apache Beam, and Apache Flink, translating single-node pipelines into multi-node clusters for accelerated processing of large-scale LLM training data.</p>
Cited references to follow up on	<p>[43] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. 2020. Overview and importance of data quality for machine learning tasks. In <i>KDD</i>. 3561–3562.</p>
Follow up Questions	<p>How does Data-Juicer specifically handle the challenges of heterogeneity in LLM data processing, and what mechanisms are in place to ensure efficient processing of diverse data formats?</p> <p>Can you provide more insights into the feedback-driven data processing aspect of Data-Juicer, particularly how the dynamic feedback loop and interactive visualization contribute to enhancing understanding and optimizing the system for LLM data processing?</p> <p>In the quantitative evaluation, what specific benchmarks and metrics were used to compare Data-Juicer against the state-of-the-art baseline, and are there any particular scenarios or use cases where Data-Juicer demonstrated superior performance or faced challenges?</p>

Notes (written with the assistance of ChatGPT)

Introduction

- Large Language Models (LLMs) achieve unprecedented intelligence, enabling applications otherwise infeasible.
- LLMs are built through pre-training on large-scale general-purpose corpus and fine-tuning with specific-purpose data.
- Challenges in LLM data processing include high heterogeneity, timely feedback, usability, customizability, and massive data volume.
- Existing open-source projects contributing LLM training data are limited, hindering quantitative understanding and enhancement.
- Data-Juicer is proposed as a one-stop data processing system addressing challenges in LLM data processing.
- Data-Juicer offers fine-grained abstraction, timely feedback, and end-to-end configurability with over 50 versatile operators and tools.
- Challenges addressed by Data-Juicer include heterogeneity, timely feedback, usability, and massive data volume.
- Data-Juicer integrates with Huggingface-datasets library, Megatron-LM, HELM, Ray, and Beam for comprehensive LLM data processing.
- Empirical evidence demonstrates Data-Juicer's ability to produce high-quality data recipes and improve LLM performance.
- Contributions include the novel system, high-quality data recipes, integration with distributed computing ecosystems, and user-centric interface designs.
- Data-Juicer's system, data recipes, and tutorials are publicly accessible on GitHub: <https://github.com/alibaba/data-juicer>.

Background and Related Works

- Language modeling is crucial for achieving machine intelligence.
- Advancements in language models, especially pre-training and fine-tuning paradigms, have led to exceptional performance in natural language processing tasks.
- The self-supervised Transformer architecture, highly parallelizable, has enabled significant increases in model parameters and training corpus scales for Large Language Models (LLMs).
- LLMs show potential for artificial general intelligence, but challenges persist in processing LLM data, whether for pre-training or fine-tuning.
- Pre-training data serves as the foundation for LLM intelligence, involving training on large amounts of high-quality data from diverse sources.
- Fine-tuning, refining pre-trained LLMs with smaller task-specific datasets, enhances capabilities and aligns models with human values.
- Challenges in processing pre-training data include filtering noise, redundancy, and toxicity, while fine-tuning data needs effective processing for maximum usefulness and minimized risks.

- The symbiotic nature of pre-training and fine-tuning data involves shared properties such as quality, diversity, and volume considerations.
- Quality is addressed through cleaning, deduplication, and detoxification in both pre-training and fine-tuning data processing.
- Diversity is emphasized by mixing various types of data to achieve appropriate diversity in both pre-training and fine-tuning data.
- The pursuit of quality and diversity often trades off with data volume, leading to challenges such as increased noise and bias.
- Existing open-source LLM data processing projects like BLOOM, PromptSource, and RedPajama have made progress but lack the abstraction and functionalities of Data-Juicer.
- Data-Juicer aims to address limitations by providing systematic and modular processing abilities for managing heterogeneous data.
- Current works lack optimal usability and the ability to explore data insight, hindering adaptability for diverse users and alternative usages.
- The focus of existing works is on functionality rather than system performance, leaving room for enhancement in efficiency, space management, and scalability.

Standardized Operator Pool

- Standardized Operator (OP) pool devised to address heterogeneity of data recipes for LLMs.
- OPs organized into four categories: Formatters, Mappers, Filters, and Deduplicators, promoting flexibility and user interaction.
- Unified Data Representation introduced through Formatter OPs, backed by Huggingface-datasets and Apache Arrow.
- Versatile Data Processing in Data-Juicer, including Mappers for text editing, Filters for conditional text filtering, and Deduplicators for dataset-level deduplication.
- Decoupling of computation for statistics and actual data processing in Filter and Deduplicator OPs for effective reuse and streamlined processes.
- Composability of Data-Juicer's OPs, allowing users to effortlessly process a variety of data types in a modular manner.
- Each OP designed for a distinct function and adaptable to different text fields, providing immense flexibility.
- Configurable parameters such as number of tokens, filtering thresholds, and auxiliary models enhance adjustability of OPs.
- OPs labeled with typical usage scenarios for easy navigation and operation, blending simplicity with power in Data-Juicer's architecture.

Feedback-Driven Data Processing

- Dynamic Feedback Loop:
 - Challenge 2 addressed with a dynamic feedback loop in the data processing pipeline.
 - Enables effective data processing and understanding through extensive visualization tools and automated tracking.

- Holistic feedback loop for LLM data processing and training with interactive visualization features.
- Hyper-Parameter Optimization (HPO) for Data Processing (Sec. 4.1):
 - Incorporation of hyper-parameter optimization (HPO) in Data-Juicer for data processing.
 - HPO tied to custom target metrics and visualization results for effective optimization.
 - Acceleration with Checkpoint and Caching:
 - Built-in checkpoint and caching management for resilient and reliable data processing.
 - Swift recovery during system restarts or failures, mitigating processing redundancy.
 - Flexibility in space-time trade-off with user-specified saving frequencies and rules.
 - Auto-HPO:
 - Automated HPO tool integrated into Data-Juicer for finding optimal data processing hyper-parameters.
 - Support for advanced HPO algorithms such as Bayesian optimization and Hyperband algorithm.
 - Users can investigate correlations and importance scores of specific hyper-parameters.
- Interactive Visualization (Sec. 4.2):
 - Interactive visualization integral to multiple feedback stages of Data-Juicer.
 - Tracer tool records sample changes after each operation for effective tracking.
 - Users can visually track the effects of individual OPs, enhancing control over the data processing.
 - Comparative visualization before and after processing, aiding in statistical analysis.
- Feedback with LLM Ecosystem Integration (Sec. 4.3):
 - Integration of rich ecosystems of LLMs to support mainstream training libraries.
 - Facilitates timely assessment of model abilities across various metrics or benchmarks.
 - Supports state-of-the-art LLM benchmarks and extension of customized evaluation benchmarks.
 - Dynamic expansion of evaluation metrics during training process for subsequent scaling predictions.
 - Leaderboard-style comparison enhances visualization of model strengths and weaknesses.
- Feedback Loop Showcase (Sec. 4.4):
 - Concrete example of the feedback loop with Data-in-the-LLMdev-Loop process.
 - Steps include analyzing the original dataset, refining parameters, processing, analyzing refined dataset, training LLMs, and comparing results.
 - Innermost loop (steps 1 and 2) for data probe creation, middle loop (steps 1~4) for refining and processing, and outermost thorough feedback loop.
 - Users can flexibly leverage built-in tools or expand the loop based on specific needs.

Boosting Usability with Built-Ins

- Unified Configuration Paradigm (Challenge 3):
 - Challenge addressed: Supporting diverse user customization preferences and technical expertise.
 - Unified and easy-to-use configuration paradigm for data recipes and extensive tools.
 - All-in-one configuration principle ensures reproducibility, traceability, and simplifies specification changes.
 - Facilitates the formation of data recipes for refinement, reuse, quantitative exploration, and automatic optimization.
- Configuring Your Data Recipe (Sec. 5.1):
 - End-to-end pipeline of data processing configurable in Data-Juicer.
 - Configuration includes processing environment parameters, OP lists, and tools.
 - Jsonargparse used for unified, flexible, and easy-to-use configuration capabilities.
 - Configuration items automatically registered for OPs and tools.
 - Users can build config files using "subtraction" or "addition" methodologies.
 - Extensive examples of pre-built data processing recipes provided for user reference.
- Dedicated Pluggable Tools (Sec. 5.2):
 - Extensible collection of powerful dedicated tools in Data-Juicer.
 - Quality Classifier:
 - Text quality classifier for culling high-quality text from heterogeneous sources.
 - Reproduced model based on GPT-3 quality scorer, expanded applicability to Chinese text and various code types.
 - Enhanced Sampler for LLM data:
 - Advanced data sampling utilities for large-scale data chunk handling in LLMs.
 - Stratified sampling technique using metadata or statistical fields for varied selection metrics.
 - Full Toolkit:
 - Includes analyzers, evaluators, and reference models, maintaining and evolving toolkit in Data-Juicer.
- User-Friendly Experiences (Sec. 5.3):
 - Designed for adaptability, catering to users with diverse expertise.
 - Zero-Code Processing:
 - Ready-to-use data recipes and plug-in tools for novice users with no advanced system knowledge.
 - Low-Code Customization:
 - Intermediate users can alter configurations, customize quality classifiers, and refine data based on pre-developed recipes.
 - Advanced Extension:
 - Experienced users can introduce new OPs, derive from base classes, and implement specific functions.
 - Decoupled design allows smooth incorporation of new tools at all stages of LLM data processing.

- Interactive Demos:
 - Series of interactive demos in Streamlit for hands-on learning and ease of adoption.

Comprehensive System Optimization

- Operator Reordering for Optimization:
 - Reorder OPs after fusion to enhance computational efficiency.
 - Utilize commutativity of Filters to prioritize less time-consuming OPs.
 - Reduces processing time and handles fewer samples for time-consuming OPs.
 - Minimizes redundant computation and overhead of initializing multiple processes.
- Optimized Space Utilization - Caching OPs and Compression:
 - Design dedicated hashing method to bypass serialization issues in OPs.
 - Enables successful caching of each OP, ensuring optimal cache management.
 - Users can activate compression technologies (Zstandard, LZ4) to reduce cache data storage.
 - Automatic compression and decompression of cache files reduce volume without compromising speed.
- Optimized Scalability - Distributed Data Processing:
 - Data-Juicer compatible with distributed processing frameworks (Ray, Apache Beam, Apache Flink).
 - Translates single-node data processing pipeline into multi-node cluster for accelerated processing.
 - Adapts HuggingFace-datasets interfaces for Ray-datasets, allowing execution in distributed mode.
 - Supports alternative back-ends like Flink, accelerating Mapper, Filter, and Deduplicator OPs in a multi-node cluster.
 - Alleviates bottlenecks on a single node caused by memory capacity and IO throughput.
 - Enhances scalability for handling large-scale LLM training data while minimizing resource requirements.

Quantitative Evaluation

- Data Quality Enhancement:
 - Data-Juicer focuses on comprehensive and flexible operability for LLM data processing.
 - Aims to generate high-quality, diverse datasets for improved LLM performance.
 - Deviates from traditional simplistic filtering for richer, more learnable information.
- Refined Pre-training Data Recipes:
 - Utilizes publicly available sources like RedPajama and the Pile for transparency.
 - Merges and enhances data quality, offering reproducible pre-training datasets.
 - LLMs pre-trained on Data-Juicer recipes consistently outperform alternatives.
 - Achieves higher performance with half the data volume compared to SOTA baselines.

- Refined Fine-tuning Data Recipes:
 - Labels subsets from Alpaca-CoT for “Instruct Fine-Tuning (IFT)” and performs data mixing.
 - LLMs trained on Data-Juicer recipes consistently demonstrate high validity.
 - Outperforms competitive fine-tuning datasets (Alpaca, Belle) with higher win rates.
 - Effectiveness confirmed compared to trivial processing strategies with higher win rates.
- System Performance and Optimization:
 - Examines end-to-end performance of Data-Juicer against RedPajama, a state-of-the-art baseline.
 - Data-Juicer demonstrates 55.6% less processing time, 63.0% less memory, and 52.2% less CPU utilization.
 - Context management, OP fusion, and reordering save up to 24.91% total time and 42.04% for fusible OPs.
- System Scalability:
 - Tests scalability on multiple servers, showcasing effective performance on Ray with up to 87.4% time reduction.
 - Limited scalability on Beam due to constraints in data loading component.
 - Demonstrates enhanced scalability and efficiency in handling large datasets for LLMs.

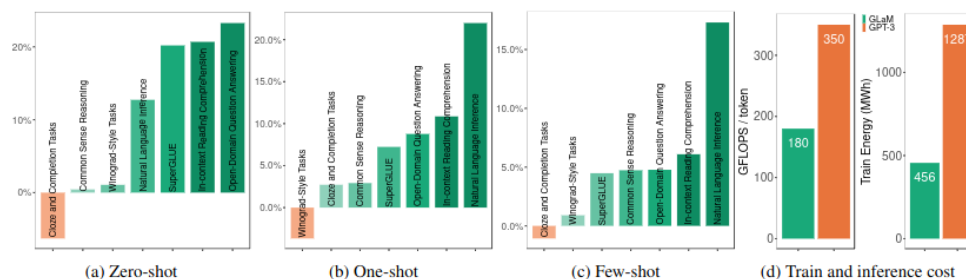
Article #16 Notes: GLaM: Efficient Scaling of Language Models with Mixture-of-Experts

Source Title	GLaM: Efficient Scaling of Language Models with Mixture-of-Experts
Source citation (APA Format)	Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., ... Cui, C. (2021). <i>GLaM: Efficient scaling of language models with mixture-of-experts</i> . (arXiv:2112.06905). arXiv. https://arxiv.org/abs/2112.06905
Original URL	https://arxiv.org/abs/2112.06905
Source type	Journal Artic
Keywords	Language Models, Efficient Scaling, Sparsely Activated MoE
#Tags	#llms, #bigdata, #moe, #efficientscaling
Summary of key points + notes (include methodology)	The article introduces the GLaM family of language models, addressing the challenges of scaling in language models like GPT-3. GLaM achieves competitive results with fewer parameters, demonstrating improved learning efficiency and lower energy consumption compared to GPT-3. The study underscores the significance of data quality in producing high-quality auto-regressive language models and highlights GLaM's ability to close the performance gap between stereotypical and anti-stereotypical examples. The GLaM models utilize sparsely activated Mixture-of-Experts (MoE) layers, providing computational flexibility and energy efficiency. The study evaluates GLaM on 29 public NLP benchmarks, showcasing its superiority in zero, one, and few-shot learning scenarios. Additionally, ethical considerations are discussed, emphasizing the potential democratization of AI usage but also addressing challenges related to biases, privacy concerns, and environmental impact. The evaluation methods include analyses of data quality effectiveness, scaling trends, and ethical challenges, with GLaM demonstrating efficiency and competitive performance.
Research Question/Problem/Need	How can the GLaM family of language models address the challenges of scaling in large language models like GPT-3, achieving competitive results with fewer parameters and demonstrating improved learning efficiency and lower energy consumption?

Important Figures

		GPT-3	GLaM	relative
cost	FLOPs / token (G)	350	180	-48.6%
	Train energy (MWh)	1287	456	-64.6%
accuracy on average	Zero-shot	56.9	62.7	+10.2%
	One-shot	61.6	65.5	+6.3%
	Few-shot	65.2	68.1	+4.4%

Comparison between GPT-3 and GLaM



An overview of the percentage change in predictive performance (higher is better) of GLaM (64B/64E) versus GPT-3 (175B) in the (a) zero-shot, (b) one-shot, and (c) few-shot setting across 7 benchmark categories with 29 public tasks in total

Dataset	Tokens (B)	Weight in mixture
Filtered Webpages	143	0.42
Wikipedia	3	0.06
Conversations	174	0.28
Forums	247	0.02
Books	390	0.20
News	650	0.02

Data and mixture weights in GLaM training set

VOCAB: (w/definition)

Pretrained Word Vectors: Word representations learned from large amounts of unlabeled text, capturing semantic relationships between words.

Contextualized Word Vectors: Word embeddings that take into account the context of a word within a sentence or document.

Mixture-of-Experts (MoE): A neural network architecture that combines multiple expert models, with a gating mechanism selecting which expert to use for a given input.

Sparse Decoder-Only Language Models: Models that focus on the decoding part of the sequence generation process and leverage sparsely activated MoE layers.

Sharding Algorithm: A technique to partition large models into smaller, manageable parts for efficient processing.

	<p>Pareto Distribution: A statistical distribution used for sampling to ensure a mix of content quality and prevent bias in the training dataset.</p> <p>Gated Linear Unit (GLU): An activation function used in place of the traditional activation function in certain model sub-layers.</p> <p>Positional Embedding: Representations added to input sequences to convey the position of each token.</p> <p>SentencePiece Subword Tokenizer: A tokenizer that breaks down words into smaller subword pieces for language modeling.</p>
<p>Cited references to follow up on</p>	<p>Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting training data from large language models. <i>CoRR</i>, abs/2012.07805, 2020.</p>
<p>Follow up Questions</p>	<p>How can the GLaM family of language models contribute to mitigating ethical challenges, such as representation bias and privacy concerns, associated with the use of large language models, and what specific strategies does the article propose for addressing these challenges?</p> <p>Given the emphasis on data quality over quantity, what are the key insights and methodologies employed in the study to ensure high-quality auto-regressive language models, and how do these insights impact the performance and efficiency of GLaM models compared to other large language models like GPT-3?</p> <p>The article highlights the efficiency of GLaM models in terms of both learning and energy consumption. Could the findings from this study inform future directions in the development of large language models, particularly in terms of achieving competitive performance with reduced computational resources, and what implications might this have for the broader field of natural language processing?</p>

Notes (written with the assistance of ChatGPT)

Introduction

- Language models have significantly contributed to the progress of natural language processing (NLP) in the last decade.
- Pretrained word vectors and contextualized word vectors have been produced using variants of language models for various NLP applications.
- The trend has shifted towards scaling with more data and larger models, allowing complex NLP tasks with less labeled data.
- Models like GPT-3 and FLAN demonstrated the feasibility of in-context learning for few-shot or zero-shot generalization, requiring very few labeled examples.
- However, further scaling is becoming expensive and energy-consuming.
- The proposed GLaM family of language models strikes a balance between dense and conditional computation, achieving competitive results with fewer parameters.
- GLaM's largest version has 1.2T parameters, with 64 experts per MoE layer, and each token activates only a subnetwork of parameters.
- GLaM outperforms GPT-3 on zero, one, and few-shot learning across 29 public NLP benchmarks, with improved learning efficiency and lower energy consumption.
- The study emphasizes the importance of data quality over quantity, even for large models, in producing high-quality auto-regressive language models.
- GLaM closes the performance gap between stereotypical and anti-stereotypical examples on the WinoGender benchmark, suggesting reduced reliance on superficial statistical correlations.
- Sparse decoder-only language models, particularly those based on MoE, show promise in achieving high-quality NLP models while saving energy costs.
- MoE is highlighted as a strong candidate for future scaling in the NLP community.

Related Work

- Language Models:
 - Neural language models, including word embedding models like word2vec and GloVe, have proven useful for various natural language processing (NLP) tasks.
- Pre-training and Fine-tuning:
 - Abundance of compute and data allows training large models through unsupervised pre-training, particularly using recurrent models and Transformers.
 - Transfer learning through pre-training and fine-tuning has demonstrated good performance on downstream tasks, but it requires task-specific fine-tuning.
- In-Context Few-shot Learning:
 - GPT-3 and related work show that scaling up language models significantly improves task-agnostic, few-shot performance without gradient updates.
- Sparsely Gated Networks:
 - Mixture-of-Experts (MoE) models, especially sparsely activated ones, offer advantages in language modeling and machine translation.

- Shazeer et al. demonstrated effective use of a large number of weights with only a small subset of the computation graph at inference time.
- Fedus et al. showcased results with a large 1 trillion parameter sparsely activated model (Switch-C), which differs from GLaM in terms of architecture and evaluation benchmarks.
- GLaM and Related Models:
 - GLaM is a family of decoder-only language models, while Switch-C is an encoder-decoder sequence-to-sequence model.
 - GLaM and Switch-C both have one trillion trainable parameters, but GLaM excels in few-shot settings without requiring fine-tuning, contrasting with Switch-C's focus on fine-tuning benchmarks like SuperGlue.
 - Table 2 provides a summary of key differences between GLaM and related models pre-trained on text corpora.

Training Dataset

- Dataset for Training:
 - A 1.6 trillion token dataset is curated, representing diverse language use cases primarily sourced from web pages.
- Quality Classification:
 - A text quality classifier is developed to distinguish between high-quality and lower-quality web content, using a feature hash-based linear classifier.
- Classifier Application:
 - The classifier rates webpages, and a Pareto distribution is used for sampling, ensuring a mix of content quality and preventing bias.
- Dataset Composition:
 - The GLaM dataset combines web pages, books, Wikipedia, forums, news, and public domain social media conversations.
- Mixture Weights:
 - Mixture weights are set based on component performance in smaller models to avoid over-sampling smaller sources like Wikipedia.
- Overlap Analysis:
 - An analysis checks for data contamination by comparing the training set with evaluation data, showing alignment with previous work (Brown et al., 2020).

Model Architecture

- Sparsely Activated MoE in GLaM Models:
 - GLaM models leverage sparsely activated Mixture-of-Experts (MoE) similar to GShard MoE Transformer.
 - Feed-forward components of every other Transformer layer are replaced with an MoE layer, each containing independent feed-forward networks as 'experts.'
- MoE Layer Structure:

- Each MoE layer has a collection of experts, and a gating function, using softmax activation, models a probability distribution over these experts.
- Sparsity is maintained, activating only a limited subset of experts for a given input token, enhancing model capacity while limiting computation.
- In this architecture, the subset size is two, and the gating network dynamically selects the best two experts for each token during inference.
- **Computational Flexibility:**
 - Each MoE layer offers $O(E^2)$ combinations of feed-forward networks, providing significantly more computational flexibility compared to the classic Transformer architecture.
- **Model Modifications:**
 - Standard positional embedding is replaced with per-layer relative positional bias.
 - In non-MoE Transformer feed-forward sub-layers, the first linear projection and activation function are replaced with the Gated Linear Unit and Gaussian Error Linear Unit.
- **Sharding Algorithm:**
 - Large GLaM models are weight- and computation-partitioned using the 2D sharding algorithm, detailed in Xu et al. (2021), for efficient processing.

Experiment Setup

- **Training Settings for GLaM:**
 - GLaM is a family of dense and sparse decoder-only language models.
 - Training variants of GLaM with hyperparameters specified for models ranging from 130 million to 1.2 trillion parameters.
- **Hyperparameter Overview:**
 - Key hyperparameters include E (number of experts), B (mini-batch size), S (input sequence length), M (model and embedding dimension), H (hidden dimension), L (number of layers), N (number of total devices), and more.
 - Dense models with comparable per-token FLOPs are included for reference.
- **Training Procedure:**
 - Common learning hyperparameters are used across all GLaM models.
 - Adafactor optimizer with specific decay schedules and threshold clipping is employed.
 - The MoE auxiliary loss is added to encourage expert load balancing.
 - Training employs SentencePiece subword tokenizer with float32 for model weights and bfloat16 for activations.
 - Larger GLaM models, such as the 64B/64E, are trained on 1,024 Cloud TPU-V4 chips.
- **Training Challenges and Strategies:**
 - Training at the trillion parameter scale is expensive and allows little room for hyperparameter tuning.
 - Smaller-scale models are trained first to expose potential issues in the dataset and infrastructure early on.

- Strategies include skipping weight updates for batches with NaNs or Infs and restarting from an earlier checkpoint in case of fluctuations or NaN/Inf occurrences during training.
- Evaluation Setting:
 - Evaluation focuses on zero, one, and few-shot learning protocols.
 - Benchmarks are chosen to match GPT-3 evaluations, with 29 datasets grouped into natural language generative (NLG) and natural language understanding (NLU) tasks.
- Benchmarks and Metrics:
 - Benchmarks include NLG tasks (e.g., TriviaQA, SQuADv2) and NLU tasks (e.g., MultiRC, COPA).
 - Metrics involve accuracy, exact match (EM), and F1 scores, with normalization for comparison.
 - Both NLG and NLU tasks are considered for the overall few-shot performance, and results are reported in a standardized format.

Results

- Extensive Evaluation of GLaM Models:
 - Evaluation of GLaM models emphasizes advantages of sparsely activated models in language modeling and their scaling trends.
 - Quantitative inspection of data quality effectiveness for language model training.
- Comparison between MoE and Dense Models:
 - GLaM (64B/64E) competes favorably with GPT-3 (175B) for zero, one, and few-shot learning.
 - Outperforms GPT-3 in 6 out of 7 categories on average, demonstrating consistent performance.
 - GLaM (64B/64E) activates fewer parameters than GPT-3, achieving similar performance with half the compute FLOPs.
- Challenging Task: TriviaQA:
 - GLaM (64B/64E) excels in the challenging open-domain question-answer task TriviaQA, surpassing GPT-3 and previous finetuned SOTA.
 - Demonstrates better one-shot performance, outperforming GPT-3 on the testing server.
- Impact of Data Quality:
 - Evaluation on the development set reveals the impact of data quality on few-shot performance.
 - Filtering text on model quality using GLaM (1.7B/64E) indicates consistent improvement in both NLG and NLU tasks, emphasizing the importance of pretraining data quality.
- Scaling Studies:
 - Scaling trends in dense models involve increasing depth and width, resulting in linear growth in parameters.
 - GLaM MoE models scale differently, growing the size or number of experts in the MoE layer.
 - MoE models consistently outperform dense models at larger scales, and additional experts enhance predictive performance.

- Efficiency of GLaM:
 - GLaM MoE models require significantly less data than dense models for similar performance in zero, one, and few-shot scenarios.
 - Computation efficiency and energy consumption analyses demonstrate that training sparsely activated models consumes much fewer computational resources than training dense models.
 - GLaM (64B/64E) achieves similar performance to GPT-3 with 1/3 of the energy cost during training. Training time and energy consumption are notably reduced, showcasing the efficiency of the MoE architecture.

Ethical Considerations

- Zero and Few-shot Inference Capabilities:
 - Large language models exhibit exciting capabilities in zero and few-shot inference.
 - Enables intuitive control of model behavior with natural language and small datasets, lowering the barrier for prototyping and application development.
 - Potential to democratize AI usage by reducing the need for specialist knowledge.
- Ethical Challenges and Considerations:
 - The versatility and power of large language models emphasize ethical challenges:
 - Representation bias.
 - Proper selection and handling of training data.
 - Documentation of training data.
 - Privacy concerns.
 - Environmental impact.
- Focus on Unintended Biases:
 - Research emphasizes unintended biases learned by language models, including correlations between gender and profession, negative sentiment about racial and religious groups, and biases related to people with disabilities.
 - Rigorous evaluation methods are essential for assessing the encoding of harmful stereotypes.
- Assessment Metrics and Methods:
 - Inspiration taken from GPT-3 for evaluation methods.
 - Examination of co-occurrence in generated text referencing identity terms.
 - Analysis of the WinoGender benchmark for coreference errors.
 - Evaluation of toxicity degeneration using the RealToxicityPrompts dataset and Perspective API.
- Co-occurrence Prompts Analysis:
 - Analysis of commonly co-occurring words in generated text for prompts related to gender, religions, racial, and ethnic identity.
 - Utilizes top-k sampling and an off-the-shelf POS tagger.
 - Results indicate associative biases and patterns in the generated text.
- WinoGender Benchmark:

- Assessment of gendered correlations and coreference errors using the WinoGender benchmark.
- GLaM (64B/64E) achieves a new state-of-the-art accuracy of 71.7%.
- Remarkably close accuracy between 'he' and 'she' examples, as well as between stereotypical and anti-stereotypical examples.
- Toxicity Degeneration Evaluation:
 - Toxicity degeneration is assessed using the RealToxicityPrompts dataset and Perspective API.
 - Relationship between Toxicity Probability of the Prompt (TPP) and the Toxicity Probability of the Continuation (TPC) is analyzed.
 - Model's TPC closely follows TPP, indicating the model's influence by the prompt.
 - Distribution of toxicity probabilities for batches of continuations is examined.
- Consideration of Ethical Challenges in Language Models:
 - Despite exciting capabilities, ethical challenges remain, requiring ongoing research and evaluation.
 - Emphasis on measurement methods and criteria for assessing general-purpose large language models.
 - The importance of assessing models on a range of metrics given their versatility and power.

Article #17 Notes: Subword Regularization: Improving Neural Network Translation Models

Source Title	Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates
Source citation (APA Format)	Kudo, T. (2018). <i>Subword regularization: Improving neural network translation models with multiple subword candidates</i> . (arXiv:1804.10959). arXiv. https://arxiv.org/abs/1804.10959
Original URL	https://arxiv.org/abs/1804.10959
Source type	Journal Article
Keywords	Neural Machine Translation (NMT), Subword Regularization, Open Vocabulary, Empirical Experiments
#Tags	#llms, #nmt, #nlp, #subwordregularization
Summary of key points + notes (include methodology)	The article discusses the challenges in Neural Machine Translation (NMT) models related to fixed word vocabularies and the resulting open vocabulary issue. It introduces Subword Regularization as a method to address this problem by breaking rare words into subword units, specifically using Byte-Pair-Encoding (BPE). The methodology involves integrating multiple segmentation candidates during on-the-fly data sampling in the training process. The experiments, conducted on various corpora, demonstrate significant improvements in translation accuracy and robustness, especially in open-domain settings. Subword regularization is highlighted as an effective approach across different NMT architectures, providing a flexible solution to enhance the performance of NMT models.
Research Question/Problem/Need	How can the accuracy and robustness of Neural Machine Translation models be enhanced in open vocabulary settings, addressing the limitations of fixed word vocabularies and increasing translation quality in the presence of unknown words?

Important Figures

Subwords (., means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_world	320 585 356 137 7 12295

Multiple subword sequences encoding the same sentence “Hello World”

Corpus	Language pair	Size of sentences			Parameters		
		train	dev	test	#vocab (Enc/Dec shared)	#dim of LSTM embedding	#layers of LSTM (Enc+Dec)
IWSLT15	en ↔ vi	133k	1553	1268	16k	512	2+2
	en ↔ zh	209k	887	1261	16k	512	2+2
IWSLT17	en ↔ fr	232k	890	1210	16k	512	2+2
	en ↔ ar	231k	888	1205	16k	512	2+2
KFTT	en ↔ ja	440k	1166	1160	8k	512	6+6
ASPEC	en ↔ ja	2M	1790	1812	16k	512	6+6
WMT14	en ↔ de	4.5M	3000	3003	32k	1024	8+8
	en ↔ cs	15M	3000	3003	32k	1024	8+8

Details of evaluation data set

Corpus	Language pair	baseline (BPE)	Proposed (one-best decoding)			Proposed (n -best decoding, $n=64$)		
			$l=1$	$l=64$ $\alpha=0.1$	$l=\infty$ $\alpha=0.2/0.5$	$l=1$	$l=64$ $\alpha=0.1$	$l=\infty$ $\alpha=0.2/0.5$
IWSLT15	en → vi	25.61	25.49	27.68*	27.71*	25.33	28.18*	28.48*
	vi → en	22.48	22.32	24.73*	26.15*	22.04	24.66*	26.31*
	en → zh	16.70	16.90	19.36*	20.33*	16.73	20.14*	21.30*
	zh → en	15.76	15.88	17.79*	16.95*	16.23	17.75*	17.29*
IWSLT17	en → fr	35.53	35.39	36.70*	36.36*	35.16	37.60*	37.01*
	fr → en	33.81	33.74	35.57*	35.54*	33.69	36.07*	36.06*
	en → ar	13.01	13.04	14.92*	15.55*	12.29	14.90*	15.36*
	ar → en	25.98	27.09*	28.47*	29.22*	27.08*	29.05*	29.29*
KFTT	en → ja	27.85	28.92*	30.37*	30.01*	28.55*	31.46*	31.43*
	ja → en	21.37	21.46	22.33*	22.04*	21.37	22.47*	22.64*
ASPEC	en → ja	40.62	40.66	41.24*	41.23*	40.86	41.55*	41.87*
	ja → en	26.51	26.76	27.08*	27.14*	27.49*	27.75*	27.89*
WMT14	en → de	24.53	24.50	25.04*	24.74	22.73	25.00*	24.57
	de → en	28.01	28.65*	28.83*	29.39*	28.24	29.13*	29.97*
	en → cs	25.25	25.54	25.41	25.26	24.88	25.49	25.38
	cs → en	28.78	28.84	29.64*	29.41*	25.77	29.23*	29.15*

Main Results (BLEU%)

Model	BLEU
Word	23.12
Character (512 nodes)	22.62
Mixed Word/Character	24.17
BPE	24.53
Unigram w/o SR ($l=1$)	24.50
Unigram w/ SR ($l=64, \alpha=0.1$)	25.04

Comparison of different segmentation algorithms (WMT14 en→de)

VOCAB: (w/definition)

Neural Machine Translation (NMT): A paradigm in machine translation that utilizes

	<p>neural networks to learn and generate translations, capturing complex linguistic patterns.</p> <p>Byte-Pair-Encoding (BPE): A subword tokenization technique that breaks down rare words into smaller units, enhancing the model's ability to handle open vocabulary.</p> <p>Recurrent Neural Network (RNN): A type of neural network designed for sequence modeling, commonly used in predicting subwords in NMT.</p> <p>Maximum Likelihood Estimation (MLE): An approach in training neural networks that maximizes the likelihood of the observed data, often used as an objective function in NMT training.</p> <p>Dropout: A regularization technique in deep learning where random units are omitted during training to prevent overfitting and enhance model generalization.</p> <p>Denosing Auto-Encoders (DAEs): Models that introduce noise to input data during training, aiming to make the model robust to variations and improve generalization.</p> <p>BLEU Score: A metric used to evaluate the quality of machine-generated translations by comparing them to reference translations, measuring precision and recall of n-grams.</p> <p>Hyperparameters: Configurable settings in a machine learning model that are not learned from the data, requiring manual tuning for optimal performance.</p> <p>Moses Tokenizer: A tool used for text tokenization, commonly employed before training subword models to preprocess data in NMT.</p> <p>Ensemble Training: A concept involving training multiple models on different subsets of data and combining their predictions to enhance overall performance.</p>
<p>Cited references to follow up on</p>	<p>Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In <i>Proc. of ACL</i>.</p>
<p>Follow up Questions</p>	<p>How does the Subword Regularization method compare to other approaches for handling open vocabulary issues in Neural Machine Translation, and are there specific scenarios where it excels or falls short?</p> <p>Given the success of Subword Regularization in improving translation accuracy and robustness, how transferable is this approach to other natural language processing tasks beyond NMT, and what considerations should be taken when applying it to different domains?</p>

	<p>The article highlights the impact of hyperparameters on the effectiveness of Subword Regularization. What further research or experimentation is needed to better understand the optimal hyperparameter settings for different corpora and language pairs, and how can these settings be generalized or fine-tuned for broader applications in NMT?</p>
--	--

Notes (written with the assistance of ChatGPT)

Introduction

- **Neural Machine Translation (NMT) Models:**
 - Operate with fixed word vocabularies (Bahdanau et al., 2014; Luong et al., 2015; Wu et al., 2016; Vaswani et al., 2017).
 - Vocabulary size crucial for training and inference.
 - Limiting vocabulary increases unknown words, affecting translation accuracy in open vocabulary settings.
- **Open Vocabulary Issue and Subword Units:**
 - Common approach: Break up rare words into subword units.
 - Examples: Byte-Pair-Encoding (BPE) (Sennrich et al., 2016).
 - BPE offers a balance between vocabulary size and decoding efficiency, addresses unknown words without special treatment.
- **BPE Segmentation and Ambiguity:**
 - BPE encodes sentences into unique subword sequences.
 - Illustration (Table 1): Multiple subword sequences encoding the same sentence, causing spurious ambiguity.
 - Variants treated as different inputs in NMT, affecting decoding.
- **Subword Regularization Method:**
 - Aim: Improve accuracy and robustness of open-vocabulary NMT.
 - Components:
 - **Integration of Multiple Segmentation Candidates:**
 - Implemented as an on-the-fly data sampling during NMT training.
 - Not specific to NMT architecture, applicable to any NMT system without changing the model structure.
 - **New Subword Segmentation Algorithm:**
 - Based on a language model.
 - Provides multiple segmentations with probabilities, emulating noise generated during actual data segmentation.
- **Empirical Experiments and Results:**
 - Multiple corpora, different sizes, and languages used.
 - Subword regularization demonstrates significant improvements over single subword sequence methods.
 - Robustness improvement shown through experiments with out-of-domain corpora.
- **Conclusion:**
 - Subword regularization emerges as an effective method for enhancing the accuracy and robustness of open-vocabulary NMT models through the integration of multiple subword segmentations during training.

Neural Machine Translation with multiple subword segmentations

- NMT Training with On-the-Fly Subword Sampling:
 - Given source sentence X and target sentence Y , segmented into subword sequences x and y .
 - NMT models translation probability $P(Y|X)$ as a sequence model generating target subwords.
 - Recurrent Neural Network (RNN) commonly used for predicting subwords.
 - Subword regularization is not limited to RNN architecture; applicable to various NMT architectures (e.g., Vaswani et al., 2017; Gehring et al., 2017).
- Standard Maximum Likelihood Estimation (MLE) Training:
 - Training objective: Maximize log-likelihood $L(\theta)$ of parallel corpus D .
 - Objective function: $L(\theta) = \sum \log P(y(s)|x(s); \theta)$ for each sentence pair in D .
 - Assumption: Source and target sentences can be segmented into multiple subword sequences with probabilities $P(x|X)$ and $P(y|Y)$.
- Subword Regularization Objective:
 - Optimize parameter set θ with marginalized likelihood (L_{marginal}) considering multiple subword segmentations.
 - Exact optimization not feasible; approximate with finite k sequences sampled from $P(x|X)$ and $P(y|Y)$.
 - Objective function: $L_{\text{marginal}}(\theta) \approx (1/k^2) \sum \sum \log P(y_j|x_i; \theta)$ for each sentence pair in D , where x_i and y_j are sampled subword sequences.
- Training Process:
 - Online training with iterative optimization of θ on mini-batches from D .
 - Subword sampling executed on-the-fly for each parameter update, yielding a good approximation of the objective function.
- Decoding in NMT:
 - For decoding, only raw source sentence X is available.
 - One-best decoding: Translate from the best segmentation x^* maximizing $P(x|X)$.
 - N-best decoding: Use n-best segmentations of $P(x|X)$ to consider multiple segmentation candidates.
 - Scoring function: $\text{score}(x, y) = \log P(y|x)/|y|^\lambda$, penalizing shorter sentences with parameter λ .
 - Parameter λ optimized with development data.
- Conclusion:
 - Describes the training process with on-the-fly subword sampling in NMT, emphasizing the flexibility of subword regularization across various NMT architectures. Additionally, it outlines decoding approaches, including one-best and n-best decoding strategies.

Related Work

- Regularization by Noise in Deep Neural Networks:

- Dropout (Srivastava et al., 2014) is a well-known example, randomly turning off hidden units during training.
- Ensemble training concept, where different models are trained on different subsets of data.
- Subword regularization considered a variant of ensemble training, introducing randomness to data inputs.
- Noise Injection in Denoising Auto-Encoders (DAEs):
 - DAEs (Vincent et al., 2008) add noise to inputs, training the model to reconstruct original inputs.
 - (Lample et al., 2017; Artetxe et al., 2017) independently propose DAEs in sequence-to-sequence learning, altering word order for compositionality.
 - Word dropout (Iyyer et al., 2015) drops words from a bag-of-words representation before averaging word embeddings.
 - (Belinkov and Bisk, 2017) explore character-based NMT with synthetic noise altering character order.
- Motivation and Similarities with Subword Regularization:
 - Subword regularization shares motivation with previous work: increasing robustness by injecting noise to input sentences.
 - Previous approaches often rely on heuristics for synthetic noises, not always reflecting real noises.
 - Subword regularization generates synthetic subword sequences using an underlying language model for better emulation of noises and segmentation errors.
 - Can be applied to both source and target sentences due to invertible conversion.
- Data Augmentation Perspective:
 - Subword regularization viewed as data augmentation.
 - Converts input sentence into multiple invariant sequences, similar to data augmentation in image classification tasks (e.g., random flipping, distorting, cropping).
- Segmentation Ambiguities in Language Modeling:
 - Latent Sequence Decompositions (LSDs) (Chan et al., 2016) marginalize over all possible segmentations, similar to subword regularization.
 - Subword regularization injects multiple segmentations with a separate language model through on-the-fly subword sampling.
 - LSDs and subword regularization handle segmentation ambiguities without assuming predetermined segmentations, with subword regularization being simple and independent of NMT architectures.
- Lattice-to-Sequence Models:
 - Extension of sequence-to-sequence models, representing input uncertainty through lattices (Su et al., 2017; Sperber et al., 2017).
 - Lattice encoded with a variant of TreeLSTM (Tai et al., 2015), requiring a change in model architecture.
 - Unlike subword regularization, lattice-to-sequence models do not handle target side ambiguities.
- Mixed Word/Character Model (Wu et al., 2016):

- Addresses out-of-vocabulary problem with a fixed vocabulary.
- Out-of-vocabulary words not collapsed into a single UNK symbol, converted into a sequence of characters with special prefixes.
- Similar to BPE, encodes a sentence into a unique fixed sequence, without considering multiple segmentations.

Experiments

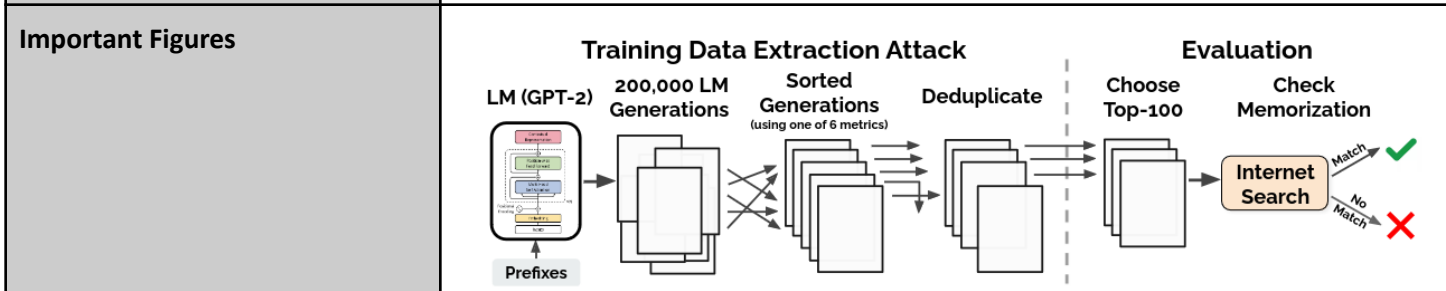
- Experiment Overview:
 - Experiments conducted using various corpora with different sizes and languages.
 - Evaluation data summarized in Table 2.
- Corpora Details:
 - IWSLT15/17 and KFTT:
 - Small corpora with diverse languages and linguistic properties.
 - Evaluate language-agnostic property of subword regularization.
 - ASPEC and WMT14 (en↔de):
 - Medium-sized corpora.
 - WMT14 (en↔cs):
 - Large corpus with over 10M parallel sentences.
- NMT System Used:
 - Implementation: GNMT (Wu et al., 2016).
 - Settings and training procedures followed (Wu et al., 2016) with adjustments based on corpus size.
- Hyperparameters:
 - Settings varied according to corpus size (see Table 2).
 - Common settings: Dropout probability 0.2, Adam and SGD for parameter estimation, length normalization, converge penalty parameters set to 0.2, decoding beam size set to 4.
- Data Preprocessing:
 - Moses tokenizer used before training subword models.
 - Chinese and Japanese processed with characters and KyTea, as Moses tokenizer does not segment sentences into words for these languages.
- Evaluation Metric:
 - Case-sensitive BLEU score (Papineni et al., 2002) used.
- Baseline System:
 - BPE segmentation used as the baseline system.
 - Three test systems evaluated with different sampling strategies.
- Sampling Strategies Evaluated:
 - Unigram language model-based subword segmentation without subword regularization ($l = 1$).
 - With subword regularization ($l = 64, \alpha = 0.1$).
 - ($l = \infty, \alpha = 0.2/0.5$) - 0.2 for IWSLT, 0.5 for others.
- Comparison and Decoding:

- One-best decoding and n-best decoding compared.
- 7 systems evaluated for each language pair.
- Main Results:
 - Table 3 shows translation experiment results.
 - Subword regularization ($l > 1$) significantly boosts BLEU scores, especially in lower resource settings (IWSLT and KFTT).
 - ($l = \infty$, $\alpha = 0.2/0.5$) slightly outperforms ($l = 64$, $\alpha = 0.1$) on IWSLT corpus.
- Results with Out-of-Domain Corpus:
 - Systems evaluated with out-of-domain in-house data (Web, patents, query logs).
 - Subword regularization achieves larger improvements (+2 points) in every domain compared to in-domain evaluations (Table 4).
 - Significant improvements even on large training datasets (WMT14), supporting the claim that subword regularization is more useful for open-domain settings.
- Comparison with Other Segmentation Algorithms:
 - Table 5 compares different segmentation algorithms.
 - Unigram language model with subword regularization achieves the best BLEU score (25.04), demonstrating the effectiveness of multiple subword segmentations.
- Impact of Sampling Hyperparameters:
 - Figure 1 shows BLEU scores for various hyperparameters on IWSLT15 (en \rightarrow vi) dataset.
 - Optimal hyperparameters depend on sampling size (l).
 - $\alpha = 0.0$ leads to performance drops, suggesting biased sampling with a language model helps emulate real noise in translation.
 - Larger l allows more aggressive regularization, more effective for low-resource settings.
 - Optimal hyperparameters challenging to determine, open question for subword sampling.
- Results with Single Side Regularization:
 - Table 6 summarizes BLEU scores with subword regularization on either source or target sentence.
 - Single side regularization has positive effects, although BLEU scores are lower than full regularization.
 - Suggests applicability of subword regularization to other NLP tasks beyond encoder-decoder architectures.

Article #18 Notes: Extracting Training Data from Large Language Models

Source Title	Extracting Training Data from Large Language Models
Source citation (APA Format)	<p>Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., & Raffel, C. (2020). <i>Extracting training data from large language models</i>. (arXiv:2012.07805). arXiv. https://arxiv.org/abs/2012.07805</p>
Original URL	https://arxiv.org/abs/2012.07805
Source type	Journal Article
Keywords	Language models, Attack methodologies, Memorization, Data privacy, Mitigation strategies
#Tags	#llms, #nlp, #bigdata, #dataprivacy, #privacyleakage
Summary of key points + notes (include methodology)	<p>The paper investigates the practical threat of extraction attacks on language models (LMs), challenging the notion that state-of-the-art LMs do not significantly memorize training examples. Demonstrating that even large LMs memorize and leak individual training examples, the study proposes a black-box query-based attack methodology to extract verbatim sequences from an LM's training set. The attacks, applicable to various LMs, including GPT-2, identify memorized content through diverse sample generation, likelihood ranking, and membership inference. The results reveal significant memorization even in LMs with minimal overfitting, highlighting the impact of model size on memorization and emphasizing the need for privacy mitigation strategies. The study employs a two-step attack process, generating text samples from the LM and predicting memorized text through membership inference. The initial approach, utilizing top-n sampling, reveals memorized content but exhibits weaknesses. The improved attack introduces enhanced sampling methods, including random sampling and conditioning on internet text, alongside refined membership inference strategies. Evaluation involves datasets with varying sample generation strategies and automated de-duplication. Manual inspection, categorization, and correlation analyses contribute to understanding the nature of memorized content and the effectiveness of different attack strategies. The study concludes with insights into mitigating privacy leakage in LMs.</p>
Research Question/Problem/	How can the threat of privacy leakage posed by language models, particularly their

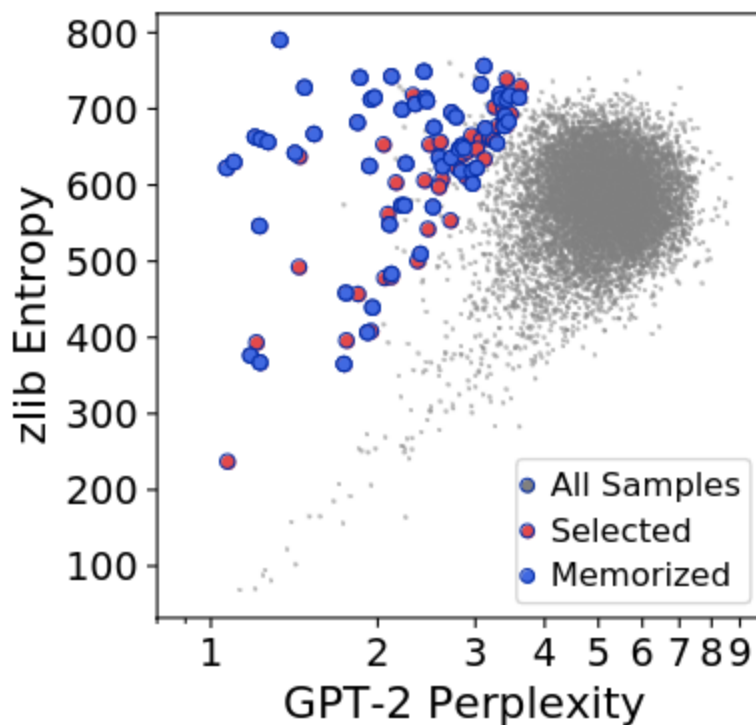
Need significant memorization of training data, be effectively addressed to ensure the responsible and secure deployment of these models in real-world applications?



Workflow of extraction attack and evaluation

Category	Count
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

Manual categorization of the 604 memorized training examples that we extract from GPT-2, along with a description of each category



The zlib entropy and the perplexity of GPT-2 XL for 200,000 samples generated with top-n sampling

Memorized String	Sequence Length	Occurrences in Data	
		Docs	Total
Y2... ██████ ...y5	87	1	10
7C... ██████ ...18	40	1	22
XM... ██████ ...WA	54	1	36
ab... ██████ ...2c	64	1	49
ff... ██████ ...af	32	1	64
C7... ██████ ...ow	43	1	83
0x... ██████ ...C0	10	1	96
76... ██████ ...84	17	1	122
a7... ██████ ...4b	40	1	311

Examples of $k = 1$ eidetic memorized, high entropy content that we extract from the training data. Each is contained in just one document. In the best case, we extract a 87-characters-long sequence that is contained in the training dataset just 10 times in total, all in the same document.

VOCAB: (w/definition)

Differential Privacy (DP): A privacy notion ensuring individual records in a training dataset remain private during model training.

	<p>DP-SGD Algorithm: A differentially private stochastic gradient descent algorithm, widely implemented for training private machine learning models.</p> <p>Top-n Sampling: A text generation method where the model samples from the top-n likely tokens.</p> <p>Membership Inference: An attack predicting if a sample was present in the training data.</p> <p>Entropy: A measure of uncertainty or disorder in a set of data.</p> <p>Beam Search: A decoding method that explores multiple possible sequences during text generation.</p> <p>Prompting: Providing input to a language model to generate desired output.</p> <p>Privacy Leakage: Unintended exposure of private information by a machine learning model.</p> <p>Memorization Spectrum (k-eidetic memorization): The range of memorization levels based on the frequency (k) of repeated instances in training data.</p> <p>Context-Dependent Memorization: The phenomenon where memorized content is influenced by the model's context during prompt input.</p> <p>Audit Models: Empirical assessment of a model's behavior, including privacy levels.</p> <p>Vetting Training Data: Scrutinizing training datasets to identify and filter sensitive content.</p>
<p>Cited references to follow up on</p>	<p>Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In <i>NAACL, 2019</i>.</p>
<p>Follow up Questions</p>	<p>How can the proposed strategies for mitigating memorization in language models be practically implemented in real-world applications, considering the trade-offs between privacy and model utility?</p> <p>In light of the demonstrated challenges in discovering memorization and the context-dependent nature of the extracted content, what additional techniques or approaches could be explored to enhance the detection and understanding of memorization in large language models?</p> <p>As language models continue to evolve and grow in size, what implications and considerations arise for privacy leakage, and what are the potential advancements</p>

in model architectures or training methodologies to address these challenges?

Notes (written with the assistance of ChatGPT)

Introduction

- Language models (LMs) are statistical models that assign probabilities to sequences of words and are crucial for various natural language processing tasks.
- Modern neural-network-based LMs use large model architectures (e.g., 175 billion parameters) and train on massive datasets (e.g., nearly a terabyte of English text).
- Scaling improves the fluency of natural language generation and allows LMs to be applied to various tasks without parameter updates.
- Machine learning models, including LMs, are known for exposing information about their training data, potentially compromising privacy.
- Privacy leakage is often associated with overfitting, where a model memorizes examples from its training set.
- State-of-the-art LMs, trained on massive de-duplicated datasets for a single epoch, exhibit little to no overfitting, leading to the assumption that they do not significantly memorize training examples.
- The paper challenges this assumption, demonstrating that large LMs do memorize and leak individual training examples.
- The proposed attack involves extracting verbatim sequences from an LM's training set using black-box query access.
- The attack generates a diverse set of high-likelihood samples, ranks them based on likelihood ratios, and identifies memorized training examples.
- The attacks apply to any language model, including those trained on sensitive and non-public data, with experiments using the GPT-2 model.
- A quantitative definition of memorization is provided, and results show that a significant percentage of candidate samples are verbatim training examples.
- The paper explores the impact of model size and string frequency on memorization and analyzes how different attack configurations extract data.
- Practical strategies to mitigate privacy leakage are discussed, including differentially-private training and recommendations for de-duplicating documents.
- While differentially-private training is theoretically effective, it can lead to longer training times and reduced utility.
- Recommendations, such as careful de-duplication of documents, are suggested to empirically mitigate memorization but may not prevent all attacks.

Background & Related Work

- Introduction to Language Models and Data Privacy Attacks:
 - Large neural network-based language models (LMs) with billion parameters play a crucial role in natural language processing.
 - Introduction to data privacy attacks in the context of LMs.
- Language Modeling:
 - Language models are fundamental for natural language processing pipelines.

- Training objectives often involve a "next-step prediction" approach.
- State-of-the-art LMs use neural networks, particularly Transformer LMs.
- Training involves minimizing the loss function over a training dataset.
- Despite the potential for memorization, LMs trained on massive datasets typically exhibit minimal memorization.
- Text Generation with Language Models:
 - LMs can generate new text by iteratively sampling tokens based on probability distributions.
 - Variations in text generation methods, including greedy sampling and top-n sampling.
 - Focus on the GPT variant of Transformer LMs, particularly GPT-2 with different model sizes.
- GPT-2 Model Details:
 - GPT-2 model family trained on data scraped from the public Internet.
 - Description of the training dataset collection process and model architecture.
 - GPT-2 does not overfit, with the training loss only around 10% smaller than the test loss.
- Training Data Privacy:
 - Undesirability of models remembering specific details about potentially private training data.
 - Overview of privacy attacks, including membership inference, model inversion, and training data extraction attacks.
 - Training data extraction attacks aim to reconstruct verbatim training examples, posing a greater risk.
- Privacy Protection Measures:
 - Discussion of differentially-private training techniques as an approach to minimize memorization.
 - Challenges associated with differentially-private mechanisms, including reduced accuracy and increased training time.
 - Notable state-of-the-art LMs, such as GPT-2, GPT-3, and T5, do not currently apply these privacy-preserving techniques.

Threat Model and Ethics

- Training Data Extraction Attacks:
 - Commonly viewed as theoretical or academic, not seen as practically exploitable.
 - Prevailing belief links privacy leakage to overfitting, and state-of-the-art LMs exhibit minimal overfitting.
 - The paper challenges this view, demonstrating the practicality of training data extraction attacks.
 - Defines "memorization" precisely and introduces the threat model and attack objectives.
- Defining Language Model Memorization:
 - Memorization is essential for LMs but needs a formal definition.
 - Introduces "eidetic memorization" as data memorized in a small set of training instances.

- Formalizes the definition with concepts of model knowledge extraction and k-eidetic memorization.
- Defines memorization as a spectrum based on the value of k and the length of memorized strings.
- Threat Model:
 - Considers an adversary with black-box access to a language model.
 - Adversary can compute probabilities of sequences but cannot inspect individual weights or hidden states.
 - Highly realistic threat model, applicable to many LMs accessible through black-box APIs.
- Risks of Training Data Extraction:
 - Discusses privacy risks, including data secrecy, contextual integrity of data, and small-k eidetic risks.
 - Focuses on small-k memorization for more impactful extraction attacks.
 - Acknowledges that LMs output memorized data even without an explicit adversary.
- Ethical Considerations:
 - Raises ethical concerns about discussing specific memorized content, especially when it contains personal information.
 - Minimizes ethical concerns by using public data and attacking a publicly available model.
 - Ethical considerations remain, and the paper masks personally-identifying information when disclosed.
 - Acknowledges potential harms and emphasizes the benefits of publicizing attacks for discussions on ethics.
 - Notes responsible disclosure efforts, including contacting individuals whose information is disclosed and informing OpenAI.

Initial Training Data Extraction Attack

- Strawman Baseline for Training Data Extraction:
 - A two-step procedure:
 - Step 1: Generate Text
 - Unconditionally sample a large quantity of data from the language model.
 - Step 2: Predict Memorized Text
 - Use a membership inference attack to remove generated samples unlikely to contain memorized text.
 - Corresponds to extracting model knowledge (Definition 1) and predicting k-eidetic memorization (Definition 2).
- Initial Text Generation Scheme (Section 4.1):
 - Initialize the language model with a one-token prompt.
 - Repeatedly sample tokens in an autoregressive fashion.
 - Sample sequences that the model considers "highly likely."
 - Concretely, sample exactly 256 tokens for each trial using the top-n strategy with $n = 40$.
- Initial Membership Inference (Section 4.2):

- Training data extraction reduces to membership inference: predict whether each sample was present in the training data.
- Use perplexity as a natural likelihood measure, measuring how well the LM predicts tokens in a sequence.
- Perplexity formula:

$$P = \exp(-1/n \sum_{i=1}^n \log f_{\theta}(x_i | x_1, \dots, x_{i-1}))$$

$$P = \exp(-1/n \sum_{i=1}^n \log f_{\theta}(x_i | x_1, \dots, x_{i-1}))$$
- Initial Extraction Results (Section 4.3):
 - Generate 200,000 samples using the GPT-2 XL model (1558M parameters) following the text generation scheme.
 - Sort samples based on model perplexity and investigate those with the lowest perplexity.
 - Baseline attack finds various memorized content, including entire licenses and popular individuals' Twitter handles or email addresses.
 - Identifies memorization but is not k-eidetic for low values of k; content likely appeared many times in the training dataset.
- Weaknesses of Initial Approach:
 - Low diversity of outputs in the sampling scheme, leading to several hundred duplicates.
 - Baseline membership inference strategy has a high number of false positives.
 - False positives often contain "repeated" strings, despite being highly unlikely, due to incorrect likelihood assignments by large LMs.

Improved Training Data Extraction Attack

- Improved Training Data Extraction:
 - Low precision and recall in the proof-of-concept attack.
 - Improved methods for sampling from the model (Section 5.1) and membership inference (Section 5.2).
- Improved Text Generation Schemes (Section 5.1):
 - Random Sampling:
 - Randomly sample from the language model.
 - Sampling With A Decaying Temperature:
 - Flatten the probability distribution with a decaying temperature.
 - Temperature starts at $t=10$ and decays to $t=1$ over the first 20 tokens.
 - Conditioning on Internet Text:
 - Seed the model with prefixes from Internet scrapes.
 - Use a subset of Common Crawl data to reduce intersection with the model's training data.
 - 50 MB dataset, 5-10 tokens sampled for context.
- Improved Membership Inference (Section 5.2):
 - Filtering Samples:
 - Low-likelihood samples filtering has poor precision.
 - Two categories of high-likelihood failures: trivial memorization and repeated substrings.

- Comparison to Other Models:
 - Use a second LM to filter samples where GPT-2's likelihood is unexpectedly high.
 - Second model trained on disjoint data or smaller GPT-2 models (Small, Medium variants).
- Comparison to zlib Compression:
 - Use zlib entropy of the text for a baseline method.
 - Ratio of GPT-2 perplexity and zlib entropy as a membership inference metric.
- Comparison to Lowercased Text:
 - Measure perplexity ratio before and after lowercasing the sequence.
- Perplexity on a Sliding Window:
 - Use the minimum perplexity averaged over a sliding window of 50 tokens to handle confidence issues in the model.

Evaluating Memorization

- Methodology (Section 6.1):
 - Three datasets of 200,000 samples (256 tokens each) using different strategies: Top-n, Temperature, Internet.
 - Ordered datasets by six membership inference metrics.
 - Selected 100 samples from the top-1000 samples for each configuration, resulting in 1,800 potential memorized content samples.
 - Applied automated fuzzy de-duplication based on trigram-multiset similarity.
- Evaluation (Section 6.1):
 - Manual inspection by four authors to determine if the sample contains memorized text.
 - Validation on the original training data obtained with limited query access to GPT-2 authors.
- Results (Section 6.2):
 - Identified 604 unique memorized training examples out of 1,800, with a 33.5% aggregate true positive rate.
 - Categorized memorized content into various types.
- Categories of Memorized Content (Section 6.2):
 - Personally Identifiable Information:
 - Individual names, phone numbers, addresses, social media accounts.
 - 46 examples of names, 32 examples of contact information.
 - URLs:
 - 50 examples of memorized URLs correctly resolving to live webpages.
 - Code:
 - 31 samples with snippets of memorized source code.
 - Unnatural Text:
 - 21 instances of random number sequences with at least 50 bits of entropy.
 - Data From Two Sources:
 - Samples containing two or more unrelated snippets of memorized text.
 - Removed Content:

- Memorized content that has been removed from the Internet.
- Extracting Longer Verbatim Sequences (Section 6.4):
 - Investigated extending the length of memorized sequences by applying a beam-search-like decoding method.
 - Extended many memorized samples to longer verbatim snippets.
- Memorization is Context-Dependent (Section 6.5):
 - Memorized content highly depends on the model's context.
 - Memorized examples are context-dependent, and the true amount of content memorized is likely underestimated due to simple prompts.
- Efficacy of Different Attack Strategies (Section 6.2):
 - Internet conditioning is the most effective for identifying memorized content.
 - All generation schemes, including baseline top-n sampling, reveal significant memorized content.
 - Comparison-based metrics (e.g., zlib) are more effective at predicting memorized content than direct LM perplexity.
 - Different extraction methods find different types of memorized content (e.g., zlib finds non-rare text, lowercase detects irregular capitalization, Small and Medium strategies find rare content).

Correlating Memorization with Model Size & Insertion Frequency

- Memorization of Naturally Occurring Canaries:
 - Language models (LMs) can memorize verbatim training strings, even with few training epochs and small train-test accuracy gaps.
 - Investigating how many times a string must appear for memorization (k in Definition 2).
 - Prior work used synthetic canaries; here, naturally occurring canaries are studied.
- Study Setup:
 - Consider a memorized content piece with a specific prefix involving a Reddit URL.
 - The URL format is located in a single document on pastebin.com, appearing multiple times in the GPT-2 training dataset.
- Methods:
 - Two approaches to extract URLs:
 - Directly prompt each GPT-2 variant with the prefix and use top-n sampling (10,000 extensions).
 - Provide GPT-2 with an additional 6-character token from the URL, use beam search for sampling.
- Results (See Table 4):
 - Difficult Approach:
 - XL (1.5 billion parameters) memorizes URLs inserted 33 times or more.
 - Medium (345 million parameters) memorizes half of the URLs.
 - Small (117 million parameters) memorizes none of the URLs.
 - Easier Approach (Additional Context and Beam Search):
 - Medium model emits four more URLs.

- Small model emits one URL inserted 359 times.
- Key Lessons:
 - Larger models memorize significantly more training data.
 - Larger model complete memorization occurs after just 33 insertions.
 - Any potentially sensitive information repeated a non-trivial amount of times is at risk for memorization, even if repeated within a single training document.

Mitigating Privacy Leakage in LMs

- Mitigation Strategies for Memorization Threats:
 - Training With Differential Privacy:
 - Differential privacy (DP) is a robust privacy notion for training ML models.
 - DP-SGD algorithms can be used for training, providing privacy guarantees.
 - Tradeoffs exist between privacy and utility; DP may limit capturing long tails of the data distribution.
 - Curating the Training Data:
 - Manual vetting of large training datasets is impractical.
 - Strategies to limit sensitive content presence, such as identifying and filtering personal information or content with restrictive terms of use.
 - Importance of careful de-duplication at more granular levels than document or paragraph to address repeated occurrences of sensitive content.
 - Careful sourcing of training data to avoid domains known for hosting sensitive content.
 - Limiting Impact on Downstream Applications:
 - Downstream applications, like dialogue systems and summarization models, often fine-tune language models on task-specific data.
 - Fine-tuning may cause the model to forget memorized data from the pre-training stage.
 - Fine-tuning could introduce its own privacy leaks if the task-specific data contains private information.
 - Future work could explore how memorization is inherited by fine-tuned models.
 - Filtering Memorized Content in Downstream Applications:
 - Downstream applications can attempt to filter out generated text containing memorized content.
 - Various membership inference strategies can be employed for reliable detection.
 - Auditing ML Models for Memorization:
 - After implementing mitigation strategies, it's crucial to audit models to empirically assess their privacy level.
 - Auditing complements theoretical upper bounds on privacy leakage.
 - Proposed methods and existing attacks can be used for model auditing.

Lessons and Future Work

- Practical Threat of Extraction Attacks:

- Prior work indicates smaller language models potentially memorize data.
- State-of-the-art LMs practically memorize training data.
- Extraction attacks are practical even with a few occurrences of a sequence.
- Our attacks, interacting as a black-box, reveal at least 604 memorized instances among 600,000 generated samples.
- This is likely a loose lower bound; more candidates could uncover additional memorized content.
- Memorization and Overfitting:
 - Common belief: preventing overfitting reduces memorization.
 - Large LMs show no significant train-test gap but still memorize verbatim examples.
 - Anomalously low losses for some training examples contribute to memorization.
 - Understanding this phenomenon is an important problem for future research.
- Impact of Model Size on Memorization:
 - Larger language models consistently memorize more training data.
 - For example, a 1.5 billion parameter GPT-2 model memorizes over 18× compared to a 124 million parameter model.
 - Privacy leakage likely to increase with the growing size of LMs.
- Challenges in Discovering Memorization:
 - Extracted training data often discovered through specific prefixes.
 - Current strategy relies on high-quality prefixes; improved selection strategies might reveal more memorized data.
- Mitigation Strategies:
 - Several directions discussed for mitigating memorization, including training with differential privacy, vetting training data, limiting downstream application impact, and auditing LMs.
 - These avenues are promising but have weaknesses; comprehensive solutions are needed for addressing memorization in modern LMs.
 - Essential to address as new generations of LMs emerge and integrate into real-world applications.

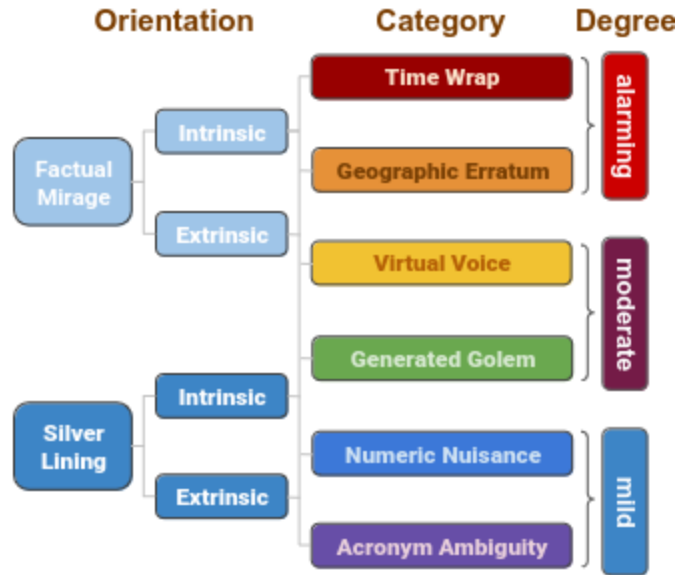
Article #19 Notes: The Troubling Emergence of Hallucination in Large Language Models – An Extensive Definition, Quantification, and Prescriptive Remediations

Source Title	The Troubling Emergence of Hallucination in Large Language Models – An Extensive Definition, Quantification, and Prescriptive Remediations
Source citation (APA Format)	Rawte, V., Chakraborty, S., Pathak, A., Sarkar, A., Tonmoy, S. M. T. I., Chadha, A., Sheth, A. P., & Das, A. (2023). <i>The troubling emergence of hallucination in large language models—An extensive definition, quantification, and prescriptive remediations</i> . (arXiv:2310.04988). arXiv. https://arxiv.org/abs/2310.04988
Original URL	https://arxiv.org/abs/2310.04988
Source type	Journal Article
Keywords	LLMs, Hallucinations, Mitigation Methods, Datasets, OpenAI
#Tags	#llms, #nlp, #hallucinations, #datasets, #mitigation
Summary of key points + notes (include methodology)	The article explores the challenges and advancements in Large Language Models (LLMs), focusing on the issue of hallucination. It introduces a comprehensive categorization of hallucination, including two primary orientations (Factual Mirage and Silver Lining) and six types, such as Acronym Ambiguity and Numeric Nuisance. A novel dataset, Hallucination eLicitation (HILT), and the Hallucination Vulnerability Index (HVI) are introduced to quantify and rank LLMs based on their susceptibility to producing hallucinations. The methodology involves annotating 75,000 text passages generated by 15 contemporary LLMs, utilizing Amazon Mechanical Turk and the MACE tool for inter-annotator agreement. The HVI calculation incorporates damping factors for comparative ranking, providing a scaled comparative spectrum. The study also explores mitigation strategies, including black-box and gray-box methods, with approaches like reranking sample responses and factuality checks. The article underscores the need for a nuanced evaluation of LLMs and their vulnerability to hallucination for informed AI-related policy-making.
Research Question/Problem/	How can the susceptibility of Large Language Models (LLMs) to hallucination be

Need

quantified and mitigated, addressing concerns and informing AI-related policy-making?

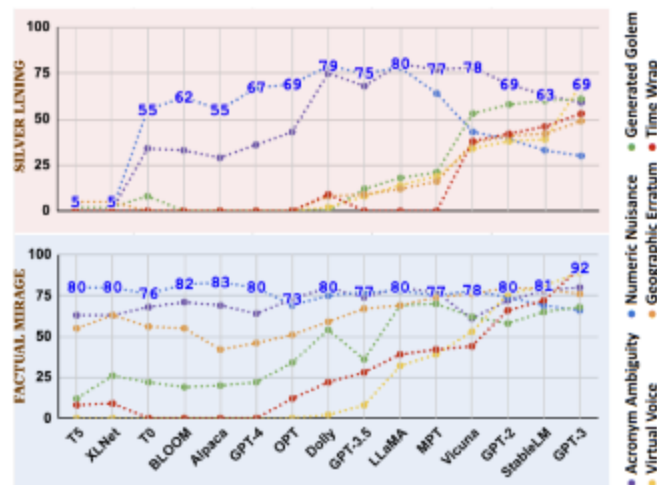
Important Figures



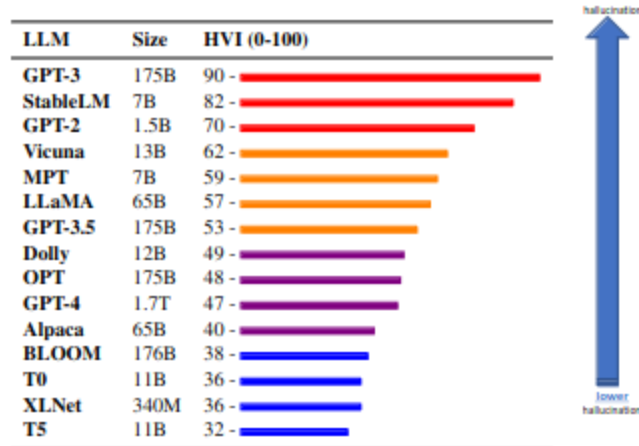
Hallucination: orientation, category, and degree

Orientation → Categories ↓	Factual Mirage (FM)		Silver Lining (SL)	
	IFM	EFM	ISL	ESL
Time Wrap	1,650	4,950	2228	3342
Acronym Ambiguity	675	550	1830	1255
Generated Golem	5,550	9,300	2302	1819
Virtual Voice	14,100	13,950	5782	8712
Numeric Nuisance	2,025	5,250	3210	5760
Geographic Erratum	6,225	6,825	1232	4530
Total	30,225	40,825	33,168	25,418

Statistics of the HILT dataset



HVI for different hallucination categories across various LLMs



The HVI scale illustrates the hallucination tendencies exhibited by various LLMs

Prompt	USA's stance on Ukraine Russia war
Hallucinated text	The official stance of the United States on the Russia-Ukraine war has been consistent in supporting Ukraine's sovereignty, territorial integrity, and the peaceful resolution of the conflict. The U.S. has condemned Russia's actions in annexing Crimea and its support for separatist movements in eastern Ukraine. Despite maintaining a diplomatic stance, the U.S. President Barack Obama stated that the United States would not deploy troops to Ukraine. Additionally, this lack of support extends to the withholding of financial and military assistance, including military equipment, training, and advisory support. This sentiment was reiterated by Paul Altman during a press review at the White House...
Mitigated text	ENTROPY _{inf} The official stance of the government on the ongoing war has been consistent in supporting resolution of the conflict. The government has condemned Russia in country and its support for separatist movements in eastern country. Despite maintaining a neutral stance, the government stated that the country would not deploy troops to unknown. Additionally, this lack of support extends to the withholding of financial and resources, including administrative, technical, and legal. This sentiment was echoed by minister during a press review at the conference...
Edit Suggestions	FACTUALITY _{cat} The official stance of the United States on the Russia-Ukraine war has been consistent in supporting Ukraine's sovereignty, territorial integrity, and the peaceful resolution of the conflict. The U.S. has condemned Russia's actions in annexing Crimea and its support for separatist movements in eastern Ukraine. Despite maintaining a diplomatic stance, U.S. President Barack Obama stated that the United States would not deploy troops to Ukraine. Additionally, this lack of support extends to the withholding of financial and military assistance, including military equipment, training, and advisory support. This sentiment was reiterated by Paul Altman during a press review at the White House...

A hallucination example pre- and post-mitigation. A - hallucinated fragments, B - high entropy fragments, C - replaced text, D - highlighted text for no information found, and E - refuted text fragments by textual entailment.

VOCAB: (w/definition)

Hallucination: The generation of misinformation or content deviating from factual accuracy by Large Language Models (LLMs).

Factual Mirage (FM): A category of hallucination orientation characterized by deviations from factual information.

Silver Lining (SL): Another category of hallucination orientation involving hallucinations related to incorrect or non-factual information.

Acronym Ambiguity: A type of hallucination involving imprecise expansions for acronyms.

Numeric Nuisance: A type of hallucination characterized by inconsistent numeric values related to past events.

Generated Golem: A type of hallucination involving the fabrication of an imaginary personality related to a past event.

Virtual Voice: A type of hallucination where quotations are generated without

	<p>sufficient evidence.</p> <p>Time Wrap: A type of hallucination characterized by the fusion of events from different timelines.</p> <p>Hallucination eLicitation (HILT) dataset: A dataset constructed for evaluating hallucination, containing 75,000 samples generated by 15 LLMs with human annotations.</p> <p>Hallucination Vulnerability Index (HVI): A comparative spectrum introduced to rank LLMs based on their vulnerability to producing hallucinations.</p>
Cited references to follow up on	<p>Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models.</p> <p>Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. <i>arXiv preprint arXiv:1907.11692</i>.</p>
Follow up Questions	<p>How can the findings from the Hallucination eLicitation dataset and the Hallucination Vulnerability Index (HVI) be practically applied to improve the safety and reliability of Large Language Models (LLMs) in real-world applications?</p> <p>Are there specific types of hallucinations, as categorized in the article, that pose higher risks or challenges in terms of misinformation and potential societal impact, and how can targeted mitigation strategies be developed to address these risks?</p> <p>Considering the diverse set of contemporary LLMs evaluated in the study, what implications do the variations in hallucination vulnerability have on the development and deployment of future LLMs, and how can the industry leverage this knowledge to enhance the responsible use of generative AI models?</p>

Notes (written with the assistance of ChatGPT)

Abstract

- Introduction to Large Language Models (LLMs):
 - Recent advancements in LLMs have gained widespread acclaim for their emerging capabilities.
 - However, the issue of hallucination has emerged as a concern, and there's a need for nuanced categorization and mitigation methods.
- Profiling Hallucination:
 - Hallucination is categorized based on degree, orientation, and category.
 - Two overarching orientations: Factual Mirage (FM) and Silver Lining (SL).
 - Further sub-categorized into intrinsic and extrinsic, with three degrees of severity (mild, moderate, alarming).
 - Six types of hallucination: Acronym Ambiguity, Numeric Nuisance, Generated Golem, Virtual Voice, Geographic Erratum, and Time Wrap.
- Dataset and Evaluation:
 - Introduction of Hallucination eLicitation dataset (75,000 samples) generated using 15 contemporary LLMs with human annotations.
 - Proposal of Hallucination Vulnerability Index (HVI) for quantifying and ranking LLMs based on their vulnerability to producing hallucinations.
- Mitigation Strategies:
 - Two proposed solution strategies for mitigating hallucinations.
 - Recognition of the risks associated with large generative AI models and the need for improved controls on hallucination.
- Industry Responses and Incidents:
 - Mention of the open letter in March 2023 calling for a moratorium on advanced AI systems.
 - The response from OpenAI emphasizing AI safety and controls on hallucination in future GPT iterations.
 - Incidents with Google's Bard leading to a market value wipeout and a lawsuit related to ChatGPT.
- Legal and Regulatory Perspectives:
 - The US Copyright Office's statement on AI-generated content lacking human authorship.
 - OpenAI's commitment to AI safety and controls in response to societal pressure.
 - Introduction of NeMo Guardrails by NVIDIA as an open-source toolkit to address hallucinations.
- Alternative Terminology and Dissatisfaction:
 - Mention of dissatisfaction within the AI community regarding the term "hallucination."
 - Prof. Christopher Manning's preference for an alternative term, and Prof. Gary Marcus's advocacy for "confabulation."

- Decision to uphold the use of the term "hallucination" in the paper.
- Positive Perspectives on Hallucination:
 - Proposal that hallucinations in LLMs could have positive implications for text summarization.
 - Examples of potential benefits of factual hallucinations in certain cases.
- Governmental Involvement and Regulatory Framework:
 - The United States and the European Union's initial proposals for the regulatory framework of AI.
 - Importance of understanding LLM vulnerability to hallucination for policymakers in assessing risks.
- Conclusion:
 - Emphasis on the need for a quantifiable spectrum (HVI) to evaluate and rank LLMs in terms of hallucination vulnerability.
 - Acknowledgment of the role of HVI in AI-related policy-making.

A Holistic View of the Hallucination Spectrum: its Types and Scales

- Background and Context:
 - Issue of hallucination explored as early as (Maynez et al., 2020).
 - Growing size of LLMs correlates with increased susceptibility to hallucination.
 - Research community shows interest in understanding and mitigating hallucination.
- Previous Research on Hallucination:
 - Early exploration of factual vs. non-factual prompts (Lee et al., 2022).
 - Categorization into intrinsic and extrinsic classes in a survey (Maynez et al., 2020).
 - Exploration of name-nationality category hallucination (Ladhak et al., 2023b).
 - Task-specific categories explored in various papers (Raunak et al., 2021; Maynez et al., 2020).
 - Preliminary examination of factual vs. non-factual prompts (Lee et al., 2022).
- General Approach to Studying Hallucination:
 - Avoidance of task-specific confinement to study hallucination nuances.
 - Emphasis on a thorough examination based on fundamental principles of text generation.
 - Findings applicable and extendable to various NLP tasks.
- Comprehensive Categorization of Hallucination:
 - Introduction of two primary orientations: Factual Mirage (FM) and Silver Lining (SL).
 - FM further subdivided into Intrinsic Factual Mirage (IFM) and Extrinsic Factual Mirage (EFM).
 - SL further divided into Intrinsic Silver Lining (ISL) and Extrinsic Silver Lining (ESL).
- Categories of Hallucination:

- Numeric Nuisance (NN): Inconsistent numeric values related to past events.
- Acronym Ambiguity (AA): Imprecise expansion for an acronym.
- Generated Golem (GG): Fabrication of an imaginary personality related to a past event.
- Virtual Voice (VV): Generation of quotations without sufficient evidence.
- Geographic Erratum (GE): Generation of incorrect location associated with an event.
- Time Wrap (TW): Fusion of events from different timelines.
- Degrees of Hallucination:
 - Annotation of hallucination degree: Mild (0), Moderate (1), Alarming (2).
 - Mild: Superficial impact.
 - Moderate: Introduction of fictitious or tangential facts.
 - Alarming: Radical dissemblance from the topic fed via the prompt.
- Visual Representation:
 - Reference to Figure 1 for a visual representation of the comprehensive categorization of hallucination.

HILT: Hallucination eLicitAtion dataset

- HILT Dataset Overview:
 - HILT is a first-of-its-kind publicly available hallucination dataset.
 - Constructed using two primary sources of data as prompts: NYTimes tweets (factually correct – FM) and Politifact dataset (factually incorrect – SL).
 - Utilized 15 LLMs to generate a total of 75,000 text passages, with each LLM producing 5,000 entries (2,500 each for FM and SL).
 - Text prompts included tweets from NYTimes and headlines from the Politifact dataset.
 - Detailed statistics about HILT provided in Table 1.
- Choice of LLMs: Rationale and Coverage:
 - Selection of 15 contemporary LLMs with exceptional results in various NLP tasks.
 - Chosen LLMs include GPT-4, GPT-3.5, GPT-3, GPT-2, MPT, OPT, LLaMA, BLOOM, Alpaca, Vicuna, Dolly, StableLM, XLNet, T5, and T0.
 - Appendix C.1 discusses additional details behind the selection criteria.
 - HVI benchmark leaderboards will remain accessible for continuous updates and contributions.
- Annotating Hallucination:
 - Amazon Mechanical Turk used for annotating the 75,000 text snippets.
 - Sentence-level annotations for hallucination orientations and categories.
 - Four annotations recorded per sentence.
 - MACE tool (Hovy et al., 2013) employed to assess inter-annotator agreement and aggregate data.

- MACE demonstrated superior performance compared to majority voting (cf. Appendix B).

Hallucination Vulnerability Index (HVI)

- Introduction to HVI:
 - With the increasing use of LLMs and their tendency to hallucinate, there is a lack of a uniform evaluation metric for measuring hallucinations.
 - Addressing this gap, the Hallucination Vulnerability Index (HVI) is introduced as a comparative spectrum for evaluating and ranking LLMs based on their vulnerability to producing hallucinations.
- HVI Calculation Equation (Eq. 1):
 - $$HVI_x = 100 / U * 2 \sum U [(N(x) - N(EFM)) * (1 - P(EFM) + \delta_1) + (N(x) - N(ESL)) * (1 - P(ESL) + \delta_2)] (1)$$
- Factors Considered in HVI Calculation:
 - U: Total number of sentences, N(x): Total number of hallucinated sentences by an LLM.
 - Consideration of the ratio of actual hallucinated sentences to the total number of sentences.
 - Capture of LLM characteristics, including higher EFM or ESL tendencies, and varying overall hallucination levels.
 - Introduction of damping factors, δ_1 and δ_2 , for comparative ranking based on $\mu \pm \text{rank}_x \times \sigma$.
 - Exclusion of variations of intrinsic hallucinations in HVI calculation due to their relatively minor impact.
- HVI Ranking and Scaling:
 - Initial calculation of HVI for all 15 LLMs with δ_1 and δ_2 as zero.
 - Mean (μ) and standard deviation (σ) calculated from initial HVIs.
 - Recalculation of HVIs for all LLMs using damping factors, resulting in a ranked and scaled comparative spectrum.
 - Scaling of HVI between 0 and 100 for ease of interpretability.
 - Similarities to z-score normalization and/or min-max normalization methods.
- Visualization:
 - Presentation of the comparative spectrum in Fig. 3.
 - HVI provides a comparative measure for ranking LLMs based on their vulnerability to hallucination.

HVI vs. LLMs size for different LLMs: An insight from HILT

- Observations on LLMs and Hallucination:
 - General observation suggests that LLMs may have a higher tendency to generate hallucinations or outputs deviating from factual information.
 - Emphasizes that the relationship between LLM size and hallucination is not a direct correlation but influenced by factors like:

- (a) Quality of training data.
 - (b) Lack of explicit training on facts.
 - (c) Overconfidence in generated responses.
- Impact of RLHF on Hallucination:
 - Noteworthy pattern: LLMs without Reinforcement Learning from Human Feedback (RLHF) tend to exhibit a higher tendency for hallucination.
 - Acknowledgment of the need for further investigation into this phenomenon in the future.
- Examination of Size's Effect on HVI:
 - Attempt to examine the effect of LLM size on HVI.
 - Observation that several other factors contribute to HVI behavior, as depicted in Fig. 6.
 - Implication that size alone does not fully explain the variation in hallucination vulnerability.

Hallucination Mitigation Strategies

- Approaches to Address Hallucination:
 - Two classes of approaches proposed:
 - (i) Preventing LLMs from hallucinating during training and/or generation.
 - (ii) Mitigating hallucination after generation.
 - (Manakul et al., 2023) introduced a classification taxonomy: black-box and gray-box methods.
 - Black-box methods involve factuality checks during/after generation without external resources.
 - Gray-box methods use external resources for factuality checks.
- Hallucination Mitigation Techniques:
 - Reranking sample responses (Dale et al., 2022).
 - Improving beam search (Sridhar and Visser, 2022).
 - Recent techniques (Li et al., 2023; Mündler et al., 2023; Pfeiffer et al., 2023; Chen et al., 2023; Zhang et al., 2023b,a; Ladhak et al., 2023a; Manakul et al., 2023; Agrawal et al., 2023) show initial attempts at reducing hallucination.
- Directions for Mitigation:
 - Exploration of two plausible directions for mitigation: automatic and human-in-the-loop.
 - Automatic (black-box): Identify high-entropy words in hallucinated text and replace them with predictions from a lower-HVI LLM.
 - Human-in-the-loop (gray-box): Sentence-level fact-checking using textual entailment techniques.
- ENTROPYBB: High Entropy Word Spotting and Replacement (Black-box):
 - Detection of high entropy words in hallucinated text by utilizing open-source LLMs.
 - Replacement of detected words with predictions from lower-HVI LLM.

- Effective strategy, particularly for hallucinations related to Generated Golem or Acronym Ambiguity.
- Lowering Concreteness of Language:
 - Proposal to substitute high entropy points with less concrete words to prevent hallucinations.
 - Concrete words are simpler to comprehend than abstract ones, based on concreteness ratings.
- FACTUALITYGB: Factuality Check of Sentences (Gray-box):
 - Utilization of Google Search API to search for a given prompt and retrieve top 20 documents.
 - Validation of each sentence of AI-generated text using RoBERTa Large for textual entailment.
 - Sentences with higher scores in refute and not enough information categories are flagged for additional human checking.
 - Empirical alert rate of 26%, implying 26% of the text requires rewriting for mitigation.
- Performance Comparison:
 - Comparative analysis of ENTROPYBB vs. FACTUALITYGB presented in Fig. 5.
 - ENTROPYBB addresses simpler hallucinations, while FACTUALITYGB handles more complex cases.
 - A balanced combination of black-box and gray-box approaches is suggested as the future avenue.

Article #20 Notes: RoBERTa: A Robustly Optimized BERT Pretraining Approach

Source Title	RoBERTa: A Robustly Optimized BERT Pretraining Approach
Source citation (APA Format)	Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). <i>RoBERTa: A robustly optimized BERT pretraining approach</i> . (arXiv:1907.11692). arXiv. https://arxiv.org/abs/1907.11692
Original URL	https://arxiv.org/abs/1907.11692
Source type	Journal Article
Keywords	RoBERTa, Language Models, Pre-training, Performance Results
#Tags	#llms, #optimization, #nlp, #bert
Summary of key points + notes (include methodology)	<p>The paper introduces and compares various self-training methods, including ELMo, GPT, BERT, XLM, and XLNet, highlighting challenges such as determining impactful aspects and computational expenses. It conducts a replication study of BERT pre-training, evaluating hyperparameter tuning and training set size effects, revealing BERT's significant undertraining. The paper introduces RoBERTa, an improved BERT model, with modifications such as longer training duration, larger batches, more extensive data, and the removal of the next sentence prediction objective. RoBERTa achieves state-of-the-art results on multiple benchmarks, presenting important design choices, training strategies, and the introduction of a novel dataset (CC-NEWS). The paper releases the RoBERTa model, pretraining, and fine-tuning code. The experimental setup involves diverse English-language corpora, and the training procedure analysis explores model configuration, dynamic masking, input formats, large batches, and text encoding. RoBERTa's configuration, evaluation setup, and results demonstrate significant improvements over BERT, emphasizing the impact of design choices. The study raises questions about the importance of model architecture and pretraining objectives compared to data size, diversity, and training time. Related work discusses pretraining objectives, common approaches, recent advances, and the importance of model size and training data. The paper's motivation is to replicate, simplify, and better tune BERT training, serving as a reference point for understanding the relative performance of pretraining methods.</p>

Research Question/Problem/Need	How can the performance of language models be improved by refining key design choices in pretraining methodologies, as demonstrated through the introduction and evaluation of RoBERTa, with an emphasis on understanding the relative impact of model architecture, pretraining objectives, and data size on downstream task performance?																																																																																																																																																							
Important Figures	<table border="1" data-bbox="542 436 1495 919"> <thead> <tr> <th>Model</th> <th>SQuAD 1.1/2.0</th> <th>MNLI-m</th> <th>SST-2</th> <th>RACE</th> </tr> </thead> <tbody> <tr> <td colspan="5"><i>Our reimplementation (with NSP loss):</i></td> </tr> <tr> <td>SEGMENT-PAIR</td> <td>90.4/78.7</td> <td>84.0</td> <td>92.9</td> <td>64.2</td> </tr> <tr> <td>SENTENCE-PAIR</td> <td>88.7/76.2</td> <td>82.9</td> <td>92.1</td> <td>63.0</td> </tr> <tr> <td colspan="5"><i>Our reimplementation (without NSP loss):</i></td> </tr> <tr> <td>FULL-SENTENCES</td> <td>90.4/79.1</td> <td>84.7</td> <td>92.5</td> <td>64.8</td> </tr> <tr> <td>DOC-SENTENCES</td> <td>90.6/79.7</td> <td>84.7</td> <td>92.7</td> <td>65.6</td> </tr> <tr> <td>BERT_{BASE}</td> <td>88.5/76.3</td> <td>84.3</td> <td>92.8</td> <td>64.3</td> </tr> <tr> <td>XLNet_{BASE} (K = 7)</td> <td>-/81.3</td> <td>85.8</td> <td>92.7</td> <td>66.1</td> </tr> <tr> <td>XLNet_{BASE} (K = 6)</td> <td>-/81.0</td> <td>85.6</td> <td>93.4</td> <td>66.7</td> </tr> </tbody> </table> <p data-bbox="521 947 1414 1010"><i>Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA</i></p> <table border="1" data-bbox="542 1045 1279 1262"> <thead> <tr> <th>bsz</th> <th>steps</th> <th>lr</th> <th>ppl</th> <th>MNLI-m</th> <th>SST-2</th> </tr> </thead> <tbody> <tr> <td>256</td> <td>1M</td> <td>1e-4</td> <td>3.99</td> <td>84.7</td> <td>92.7</td> </tr> <tr> <td>2K</td> <td>125K</td> <td>7e-4</td> <td>3.68</td> <td>85.2</td> <td>92.9</td> </tr> <tr> <td>8K</td> <td>31K</td> <td>1e-3</td> <td>3.77</td> <td>84.6</td> <td>92.8</td> </tr> </tbody> </table> <p data-bbox="521 1289 1471 1352"><i>Perplexity on held-out training data (ppl) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (bsz)</i></p> <table border="1" data-bbox="542 1388 1495 1843"> <thead> <tr> <th>Model</th> <th>data</th> <th>bsz</th> <th>steps</th> <th>SQuAD (v1.1/2.0)</th> <th>MNLI-m</th> <th>SST-2</th> </tr> </thead> <tbody> <tr> <td colspan="7">RoBERTa</td> </tr> <tr> <td>with BOOKS + WIKI</td> <td>16GB</td> <td>8K</td> <td>100K</td> <td>93.6/87.3</td> <td>89.0</td> <td>95.3</td> </tr> <tr> <td>+ additional data (§3.2)</td> <td>160GB</td> <td>8K</td> <td>100K</td> <td>94.0/87.7</td> <td>89.3</td> <td>95.6</td> </tr> <tr> <td>+ pretrain longer</td> <td>160GB</td> <td>8K</td> <td>300K</td> <td>94.4/88.7</td> <td>90.0</td> <td>96.1</td> </tr> <tr> <td>+ pretrain even longer</td> <td>160GB</td> <td>8K</td> <td>500K</td> <td>94.6/89.4</td> <td>90.2</td> <td>96.4</td> </tr> <tr> <td colspan="7">BERT_{LARGE}</td> </tr> <tr> <td>with BOOKS + WIKI</td> <td>13GB</td> <td>256</td> <td>1M</td> <td>90.9/81.8</td> <td>86.6</td> <td>93.7</td> </tr> <tr> <td colspan="7">XLNet_{LARGE}</td> </tr> <tr> <td>with BOOKS + WIKI</td> <td>13GB</td> <td>256</td> <td>1M</td> <td>94.0/87.8</td> <td>88.4</td> <td>94.4</td> </tr> <tr> <td>+ additional data</td> <td>126GB</td> <td>2K</td> <td>500K</td> <td>94.5/88.8</td> <td>89.8</td> <td>95.6</td> </tr> </tbody> </table>	Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE	<i>Our reimplementation (with NSP loss):</i>					SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2	SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0	<i>Our reimplementation (without NSP loss):</i>					FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8	DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6	BERT _{BASE}	88.5/76.3	84.3	92.8	64.3	XLNet _{BASE} (K = 7)	-/81.3	85.8	92.7	66.1	XLNet _{BASE} (K = 6)	-/81.0	85.6	93.4	66.7	bsz	steps	lr	ppl	MNLI-m	SST-2	256	1M	1e-4	3.99	84.7	92.7	2K	125K	7e-4	3.68	85.2	92.9	8K	31K	1e-3	3.77	84.6	92.8	Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2	RoBERTa							with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3	+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6	+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1	+ pretrain even longer	160GB	8K	500K	94.6/89.4	90.2	96.4	BERT _{LARGE}							with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7	XLNet _{LARGE}							with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4	+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6
Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE																																																																																																																																																				
<i>Our reimplementation (with NSP loss):</i>																																																																																																																																																								
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2																																																																																																																																																				
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0																																																																																																																																																				
<i>Our reimplementation (without NSP loss):</i>																																																																																																																																																								
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8																																																																																																																																																				
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6																																																																																																																																																				
BERT _{BASE}	88.5/76.3	84.3	92.8	64.3																																																																																																																																																				
XLNet _{BASE} (K = 7)	-/81.3	85.8	92.7	66.1																																																																																																																																																				
XLNet _{BASE} (K = 6)	-/81.0	85.6	93.4	66.7																																																																																																																																																				
bsz	steps	lr	ppl	MNLI-m	SST-2																																																																																																																																																			
256	1M	1e-4	3.99	84.7	92.7																																																																																																																																																			
2K	125K	7e-4	3.68	85.2	92.9																																																																																																																																																			
8K	31K	1e-3	3.77	84.6	92.8																																																																																																																																																			
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2																																																																																																																																																		
RoBERTa																																																																																																																																																								
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3																																																																																																																																																		
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6																																																																																																																																																		
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1																																																																																																																																																		
+ pretrain even longer	160GB	8K	500K	94.6/89.4	90.2	96.4																																																																																																																																																		
BERT _{LARGE}																																																																																																																																																								
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7																																																																																																																																																		
XLNet _{LARGE}																																																																																																																																																								
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4																																																																																																																																																		
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6																																																																																																																																																		

Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps)

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
<i>Single-task single models on dev</i>										
BERT _{LARGE}	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet _{LARGE}	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	90.2/90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4	91.3	-
<i>Ensembles on test (from leaderboard as of July 25, 2019)</i>										
ALICE	88.2/87.9	95.7	90.7	83.5	95.2	92.6	68.6	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5

Results on GLUE. All results are based on a 24-layer architecture. BERT_{LARGE} and XLNet_{LARGE} results are from Devlin et al. (2019) and Yang et al. (2019), respectively

Model	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
<i>Single models on dev, w/o data augmentation</i>				
BERT _{LARGE}	84.1	90.9	79.0	81.8
XLNet _{LARGE}	89.0	94.5	86.1	88.8
RoBERTa	88.9	94.6	86.5	89.4
<i>Single models on test (as of July 25, 2019)</i>				
XLNet _{LARGE}			86.3 [†]	89.1 [†]
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			87.0[†]	89.9[†]

Results on SQuAD. † indicates results that depend on additional external training data

Model	Accuracy	Middle	High
<i>Single models on test (as of July 25, 2019)</i>			
BERT _{LARGE}	72.0	76.6	70.1
XLNet _{LARGE}	81.7	85.4	80.2
RoBERTa	83.2	86.5	81.3

Results on the RACE test set

VOCAB: (w/definition)

Self-training methods: Techniques for training language models, including ELMo, GPT, BERT, XLM, and XLNet.

	<p>BERT (Bidirectional Encoder Representations from Transformers): A pivotal self-training method with bidirectional context representations and masked language modeling.</p> <p>RoBERTa: An enhanced version of BERT, incorporating modifications like longer training, larger batches, and the removal of next sentence prediction.</p> <p>GLUE (General Language Understanding Evaluation): A benchmark comprising nine datasets to evaluate natural language understanding systems.</p> <p>SQuAD (Stanford Question Answering Dataset): A benchmark task requiring models to answer questions based on provided contexts.</p> <p>RACE (ReAding Comprehension from Examinations): A comprehensive reading comprehension dataset, particularly focused on English examinations in China.</p> <p>Byte-Pair Encoding (BPE): A text compression technique used in language models, serving as a hybrid between character- and word-level representations.</p> <p>Mixed precision floating-point arithmetic: A computation technique utilizing both low- and high-precision floating-point numbers for efficiency in the training process.</p> <p>Finetuning: The process of adapting a pre-trained model to a specific task or dataset.</p> <p>Model architecture: The structure and design of a neural network, crucially explored in the context of BERT and RoBERTa.</p>
<p>Cited references to follow up on</p>	<p>Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In <i>North American Association for Computational Linguistics (NAACL)</i>.</p>
<p>Follow up Questions</p>	<p>Exploration of Training Efficiency: How do the modifications introduced in RoBERTa, such as longer training durations and larger batches, impact the efficiency of the training process, and what are the potential trade-offs in terms of computational resources?</p> <p>Generalization to Other Languages: Given the focus on the CC-NEWS dataset and English-language corpora, how well does RoBERTa generalize to languages other than English, and what considerations should be taken into account for cross-lingual applications?</p> <p>Fine-tuning Strategies: The article mentions task-specific finetuning approaches for</p>

	<p>certain benchmarks. Could further insights be provided into the specific strategies employed for tasks like QNLI and WNLI, and how transferable are these strategies to other diverse tasks or domains?</p>
--	--

Notes (written with the assistance of ChatGPT)

Introduction

- Self-training methods:
 - ELMo (Peters et al., 2018)
 - GPT (Radford et al., 2018)
 - BERT (Devlin et al., 2019)
 - XLM (Lample and Conneau, 2019)
 - XLNet (Yang et al., 2019)
- Challenges with self-training methods:
 - Difficulty in determining the most impactful aspects
 - Computational expense of training
 - Limited tuning due to computational constraints
 - Use of private training data with varying sizes
- Replication study of BERT pre-training:
 - Evaluation of hyperparameter tuning and training set size effects
 - Observation of BERT being significantly undertrained
- Introduction of RoBERTa:
 - Improved training recipe for BERT models
 - Modifications:
 1. Longer training duration
 2. Larger batches
 3. More extensive data
 4. Removal of next sentence prediction objective
 5. Training on longer sequences
 6. Dynamic masking pattern on training data
- Dataset:
 - Collection of a new dataset (CC-NEWS) comparable in size to privately used datasets
 - Aims to better control for training set size effects
- Performance results:
 - RoBERTa achieves a score of 88.5 on the public GLUE leaderboard
 - Establishes a new state-of-the-art on 4/9 GLUE tasks: MNLI, QNLI, RTE, and STS-B
 - Matches or exceeds the performance of post-BERT methods
 - Matches state-of-the-art results on SQuAD and RACE
- Contributions of the paper:
 - Presentation of important BERT design choices and training strategies
 - Introduction of alternatives leading to improved downstream task performance
 - Use of a novel dataset, CC-NEWS, confirming that more data for pre-training enhances performance on downstream tasks

- Demonstration that under certain design choices, masked language model pretraining (BERT) is competitive with recently published methods
- Release:
 - The paper releases the model, pretraining, and fine-tuning code implemented in PyTorch (Paszke et al., 2017).

Background

- Overview of BERT Pretraining Approach:
 - BERT (Devlin et al., 2019) pretraining approach discussed
 - Subsequent examination of training choices outlined
- Setup (Input Representation):
 - Input consists of concatenation of two segments (sequences of tokens), x_1, \dots, x_N and y_1, \dots, y_M
 - Segments presented as a single input sequence to BERT with special tokens ([CLS], [SEP], [EOS])
 - Constraint: $M + N < T$, where T is the maximum sequence length during training
- Architecture:
 - BERT utilizes the transformer architecture (Vaswani et al., 2017)
 - Transformer architecture with L layers, each block with A self-attention heads and hidden dimension H
- Training Objectives:
 - Two objectives during pretraining: masked language modeling (MLM) and next sentence prediction (NSP)
 - Masked Language Model (MLM):
 - Randomly selects 15% of input tokens for possible replacement
 - Replaces 80% with [MASK], 10% left unchanged, and 10% replaced by a randomly selected vocabulary token
 - Original implementation involves one-time random masking, but in practice, data is duplicated to vary the mask for each training sentence
 - Next Sentence Prediction (NSP):
 - Binary classification loss predicting whether two segments follow each other in the original text
 - Positive examples from consecutive sentences, negative examples from segments of different documents
 - Positive and negative examples sampled with equal probability
 - NSP objective designed to improve downstream tasks requiring reasoning about relationships between pairs of sentences
- Optimization:
 - Optimized with Adam (Kingma and Ba, 2015) using specific parameters
 - Learning rate warmed up over the first 10,000 steps, then linearly decayed
 - Training with a dropout of 0.1 on all layers and attention weights, GELU activation function

- Pretraining for $S = 1,000,000$ updates, mini-batches with $B = 256$ sequences of maximum length $T = 512$ tokens
- Data:
 - BERT trained on a combination of BOOKCORPUS (Zhu et al., 2015) and English WIKIPEDIA
 - Totaling 16GB of uncompressed text used for training

Experimental Setup

- Implementation:
 - BERT reimplementaion in FAIRSEQ (Ott et al., 2019)
 - Primarily follows original BERT optimization hyperparameters
 - Tuning of peak learning rate, warmup steps, Adam epsilon term, and β_2 for stability with large batch sizes
 - Pretraining with sequences of at most $T = 512$ tokens
 - Utilizes mixed precision floating-point arithmetic on DGX-1 machines with 8×32 GB Nvidia V100 GPUs interconnected by Infiniband
- Data:
 - Five English-language corpora used for BERT-style pretraining, totaling over 160GB of uncompressed text
 - Text corpora include:
 - BOOKCORPUS plus English WIKIPEDIA (16GB)
 - CC-NEWS from CommonCrawl News dataset (76GB after filtering)
 - OPENWEBTEXT, an open-source recreation of WebText corpus (38GB)
 - STORIES, a dataset filtered to match story-like style (31GB)
- Evaluation:
 - Evaluation on downstream tasks using three benchmarks
 - GLUE (General Language Understanding Evaluation):
 - Collection of 9 datasets for evaluating natural language understanding systems
 - Tasks framed as either single-sentence or sentence-pair classification
 - Evaluation on development sets after finetuning pretrained models on corresponding single-task training data
 - Public leaderboard used for test set results in Section 5, with task-specific modifications described
 - SQuAD (Stanford Question Answering Dataset):
 - Provides a paragraph of context and a question
 - Task is to answer the question by extracting the relevant span from the context
 - Evaluation on two versions: V1.1 (context always contains an answer) and V2.0 (some questions not answered in the provided context)

- Different prediction methods for V1.1 and V2.0
 - RACE (ReAding Comprehension from Examinations):
 - Large-scale reading comprehension dataset with over 28,000 passages and nearly 100,000 questions
 - Collected from English examinations in China for middle and high school students
 - Each passage associated with multiple questions, requiring selection of one correct answer from four options
 - Significantly longer context than other popular reading comprehension datasets, with a high proportion of questions requiring reasoning

Training Procedure Analysis

- Model Configuration:
 - Fixed model architecture similar to BERTBASE (L = 12, H = 768, A = 12, 110M params)
- Static vs. Dynamic Masking (Section 4.1):
 - Original BERT used static masking with duplicated data for varied masking
 - Dynamic masking introduced, generating the masking pattern every time a sequence is fed to the model
 - Dynamic masking comparable or slightly better than static masking, chosen for efficiency benefits in subsequent experiments
- Model Input Format and Next Sentence Prediction (Section 4.2):
 - Comparison of different training formats:
 - SEGMENT-PAIR+NSP: Original BERT input format with NSP loss
 - SENTENCE-PAIR+NSP: Each input contains a pair of natural sentences with NSP loss
 - FULL-SENTENCES: Each input packed with full sentences, NSP loss removed
 - DOC-SENTENCES: Similar to FULL-SENTENCES but may not cross document boundaries, NSP loss removed
 - Results show variations in performance; FULL-SENTENCES chosen for subsequent experiments for easier comparison
- Training with Large Batches (Section 4.3):
 - Training with larger batches explored for optimization speed and end-task performance improvement
 - Comparison of perplexity and end-task performance with increasing batch sizes
 - Larger batches improve perplexity and end-task accuracy, also easier for parallelization
 - Training with batches of 8K sequences chosen for later experiments
- Text Encoding (Section 4.4):

- Introduction of Byte-Pair Encoding (BPE) as a hybrid between character- and word-level representations
- Radford et al. (2019) introduced byte-level BPE vocabulary as an alternative
- Original BERT used character-level BPE vocabulary of size 30K
- Consideration of training BERT with a larger byte-level BPE vocabulary containing 50K subword units
- Slight differences observed, but universal encoding scheme preferred for advantages, chosen for subsequent experiments

RoBERTa

- RoBERTa Configuration:
 - Trained with dynamic masking (Section 4.1)
 - Utilizes FULL-SENTENCES without NSP loss (Section 4.2)
 - Large mini-batches employed (Section 4.3)
 - Larger byte-level BPE used (Section 4.4)
- Evaluation Setup:
 - Model architecture follows BERTLARGE (L = 24, H = 1024, A = 16, 355M parameters)
 - Pretraining conducted for 100K steps over a BOOKCORPUS plus WIKIPEDIA dataset
 - Exploration of the impact of data size and diversity, training passes, and longer pretraining durations
- Results (Table 4):
 - RoBERTa shows significant improvement over BERTLARGE, affirming the importance of the explored design choices
 - Further improvements observed with increased data size and diversity
 - Longer pretraining durations (300K and 500K steps) result in significant gains without apparent overfitting
- Evaluation on GLUE, SQuAD, and RACE (Sections 5.1 to 5.3):
 - GLUE Benchmark:
 - RoBERTa achieves state-of-the-art results on all 9 GLUE task development sets
 - Outperforms both BERTLARGE and XLNetLARGE, questioning the importance of architecture and pretraining objective
 - Submission to GLUE leaderboard yields state-of-the-art results on 4 out of 9 tasks and highest average score
 - SQuAD Benchmark:
 - Finetuning on SQuAD conducted only with provided training data, simplifying the approach
 - RoBERTa matches state-of-the-art on SQuAD v1.1 and sets a new state-of-the-art on SQuAD v2.0

- Top-performing among systems not relying on data augmentation in the public SQuAD 2.0 leaderboard
 - RACE Benchmark:
 - RoBERTa achieves state-of-the-art results on both middle-school and high-school settings in RACE test sets
- Task-Specific Modifications (Sections 5.1 and 5.2):
 - Task-specific finetuning approaches adopted for QNLI and WNLI tasks to align with competitive leaderboard results
- Overall Implications:
 - RoBERTa consistently outperforms its BERT-based counterparts, emphasizing the impact of modifications in the pretraining procedure
 - Questions raised about the relative importance of model architecture and pretraining objective compared to data size, diversity, and training time.

Related Work

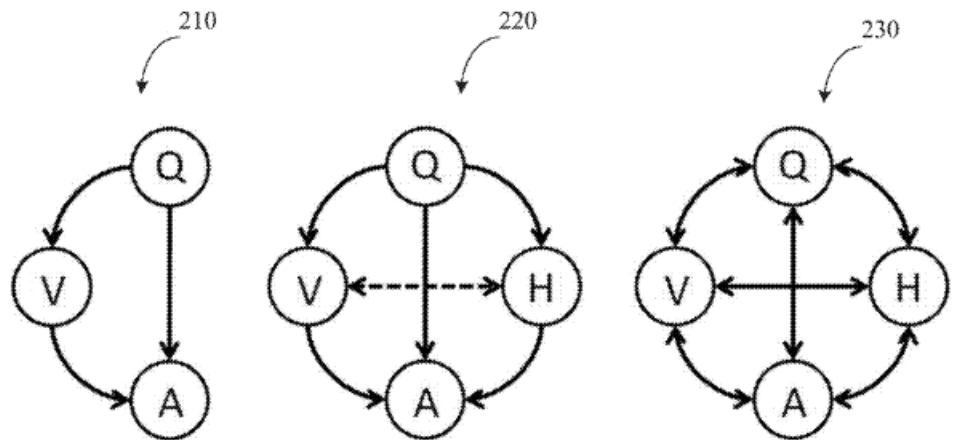
- Overview of Pretraining Objectives:
 - Different pretraining objectives employed, including language modeling, machine translation, and masked language modeling.
 - Language modeling sources include works by Dai and Le (2015), Peters et al. (2018), Howard and Ruder (2018).
 - Machine translation used as a pretraining objective (McCann et al., 2017).
 - Introduction of masked language modeling in works like Devlin et al. (2019) and Lample and Conneau (2019).
- Common Approaches in Recent Papers:
 - Common practice involves finetuning models for specific end tasks after pretraining (Howard and Ruder, 2018; Radford et al., 2018).
 - Pretraining often involves variants of masked language model objectives.
- Recent Advances in Pretraining Methods:
 - Recent methods have demonstrated improved performance through:
 - Multi-task fine-tuning (Dong et al., 2019).
 - Incorporating entity embeddings (Sun et al., 2019).
 - Introducing span prediction (Joshi et al., 2019).
 - Exploration of autoregressive pretraining variants (Song et al., 2019; Chan et al., 2019; Yang et al., 2019).
- Importance of Model Size and Training Data:
 - Consistent performance improvement observed by training larger models on more data (Devlin et al., 2019; Baevski et al., 2019; Yang et al., 2019; Radford et al., 2019).
- Motivation for the Study:
 - Goal is to replicate, simplify, and better tune the training of BERT.

- Serves as a reference point for understanding the relative performance of various pretraining methods.

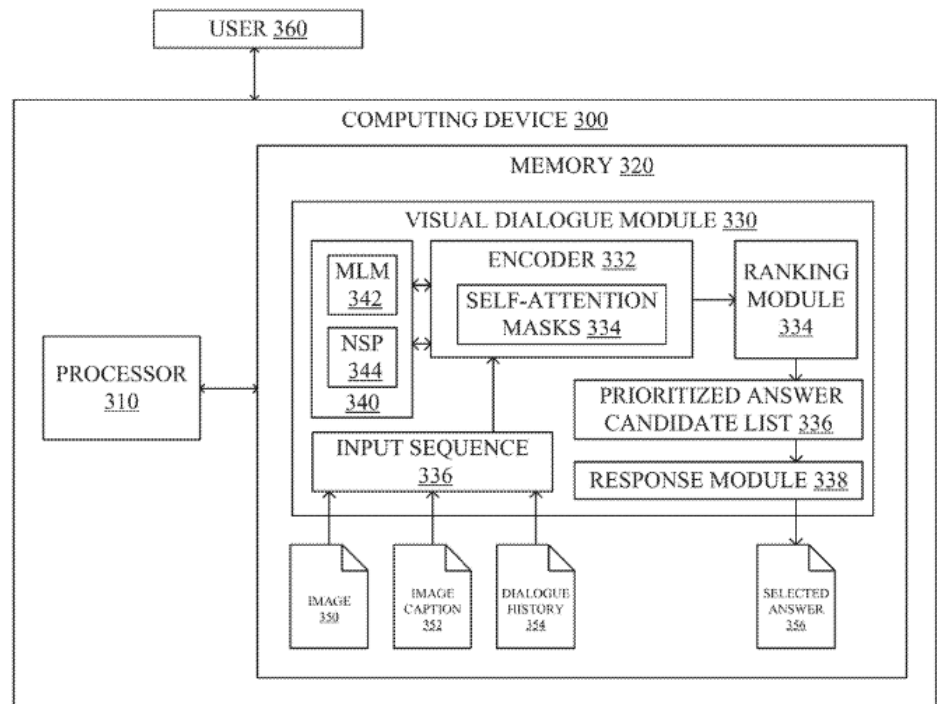
Article #21 Notes: Unified Vision and Dialogue Transformer with BERT

Source Title	Unified Vision and Dialogue Transformer with BERT
Source citation (APA Format)	Wang, Y., Hoi, C. H., & Joty, S. R. (2021, July 29). <i>Unified Vision and Dialogue Transformer with BERT</i> . Google Patents. https://patents.google.com/patent/US20210232773A1/en
Original URL	https://patents.google.com/patent/US20210232773A1/en
Source type	Patent
Keywords	Visual Dialogue Model, Transformer Encoder Network, Unified Contextualized Representation, Encoded Visual Dialogue Input, Self-Attention Mask
#Tags	#nlp, #transformers, #bert, #multimodal, #attention
Summary of key points + notes (include methodology)	A unified transformer vision and dialogue BERT (unidirectional transformer encoder) is proposed. The model receives an image and text input, including a dialogue history between the model and a human user. The model then generates an encoded visual dialogue input, which includes a position level encoding and a segment level encoding. The model further generates a unified contextualized representation using a transformer encoder network. Finally, the model generates an answer prediction using a first self-attention mask associated with discriminative settings of the transformer encoder network or a second self-attention mask associated with generative settings of the transformer encoder network.
Research Question/Problem/Need	How can a unified transformer vision and dialogue BERT (unidirectional transformer encoder) be used for answering questions about images and dialogue?

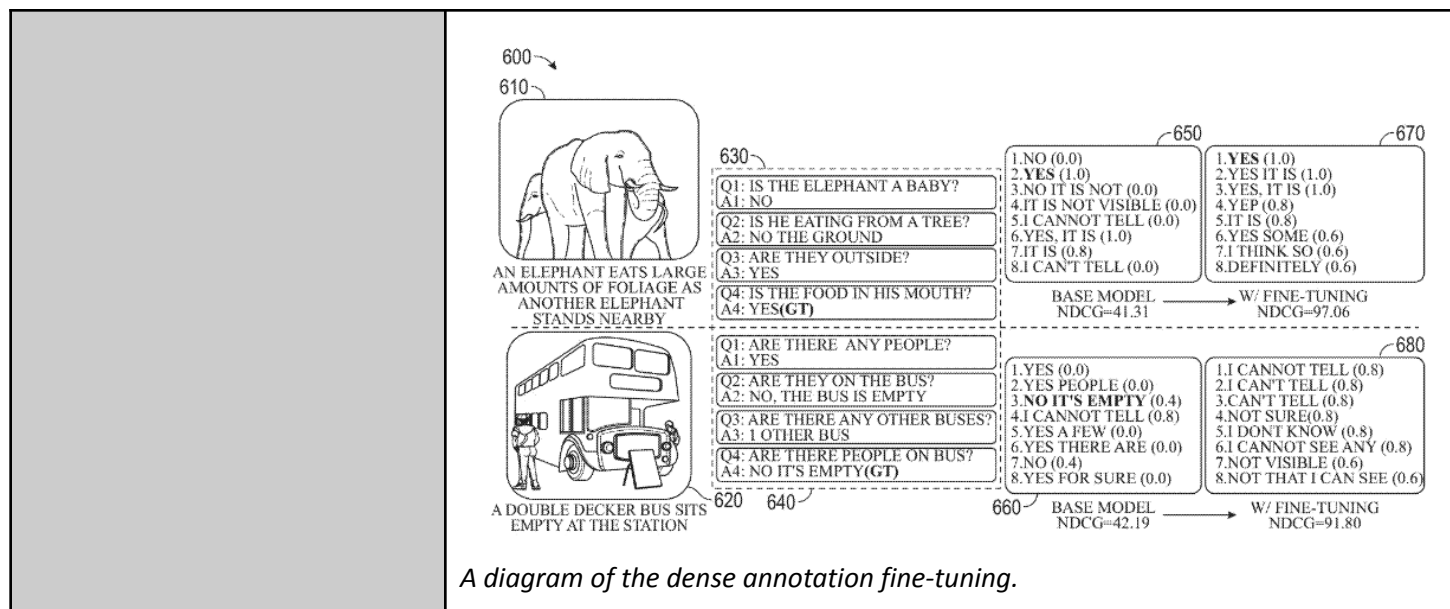
Important Figures



A diagram of the unified transformer vision and dialogue BERT model



A diagram of the ranking module



A diagram of the dense annotation fine-tuning.

<p>VOCAB: (w/definition)</p>	<p>Transformer Vision and Dialogue BERT: A neural network model that jointly encodes visual and linguistic information to answer questions about images and dialogue</p> <p>Unified Contextualized Representation: A representation of the image and dialogue that captures the relationships between the different elements of the input</p> <p>Position Level Encoding: An encoding that represents the position of each word in the input sequence</p> <p>Segment Level Encoding: An encoding that represents whether each word belongs to the image input or the dialogue input</p> <p>Self-Attention Mask: A mask that is used to control the attention of the transformer encoder</p> <p>Discriminative Settings: Settings of the transformer encoder that are used to answer questions</p> <p>Generative Settings: Settings of the transformer encoder that are used to generate text</p> <p>Visual Encoder: A component of the model that encodes the image input</p> <p>Dialogue Encoder: A component of the model that encodes the dialogue input</p> <p>Response Prediction: The process of generating an answer to a question</p>
-------------------------------------	---

<p>Cited references to follow up on</p>	<p>There were no cited references throughout the patent</p>
--	---

Follow up Questions	<p>What are some potential applications of this model beyond answering questions about images and dialogue? This question encourages broader thinking about the model's capabilities and potential impact.</p> <p>How does this model compare to other approaches to visual question answering? This question invites a deeper exploration of the patent's innovation and positioning within the field.</p> <p>What are the ethical considerations involved in developing and using models that can generate text? This question prompts reflection on the broader implications of the technology, which is particularly valuable given the potential for misuse of AI-generated language.</p>
----------------------------	--

Notes (written with the assistance of ChatGPT)

Background

- Visual dialogue systems aim to enable natural and engaging conversations about images.
- These systems typically combine visual and linguistic information to provide informative and relevant responses to user queries.
- Traditional visual dialogue systems have relied on separate visual and linguistic models, which can limit their ability to effectively understand and respond to complex queries.
- The proposed unified transformer vision and dialogue BERT model addresses these limitations by jointly encoding visual and linguistic information using a single transformer encoder.

Description of the Invention

- The proposed unified transformer vision and dialogue BERT model receives an image and text input, including a dialogue history between the model and a human user.
- The model then generates an encoded visual dialogue input, which includes a position level encoding and a segment level encoding.
- The model further generates a unified contextualized representation using a transformer encoder network.
- Finally, the model generates an answer prediction using a first self-attention mask associated with discriminative settings of the transformer encoder network or a second self-attention mask associated with generative settings of the transformer encoder network.

Examples

- The model can be used to answer questions about images, such as "What is the color of the dog in the image?".
- The model can also be used to follow instructions in dialogue, such as "Describe the image to me."

Advantages

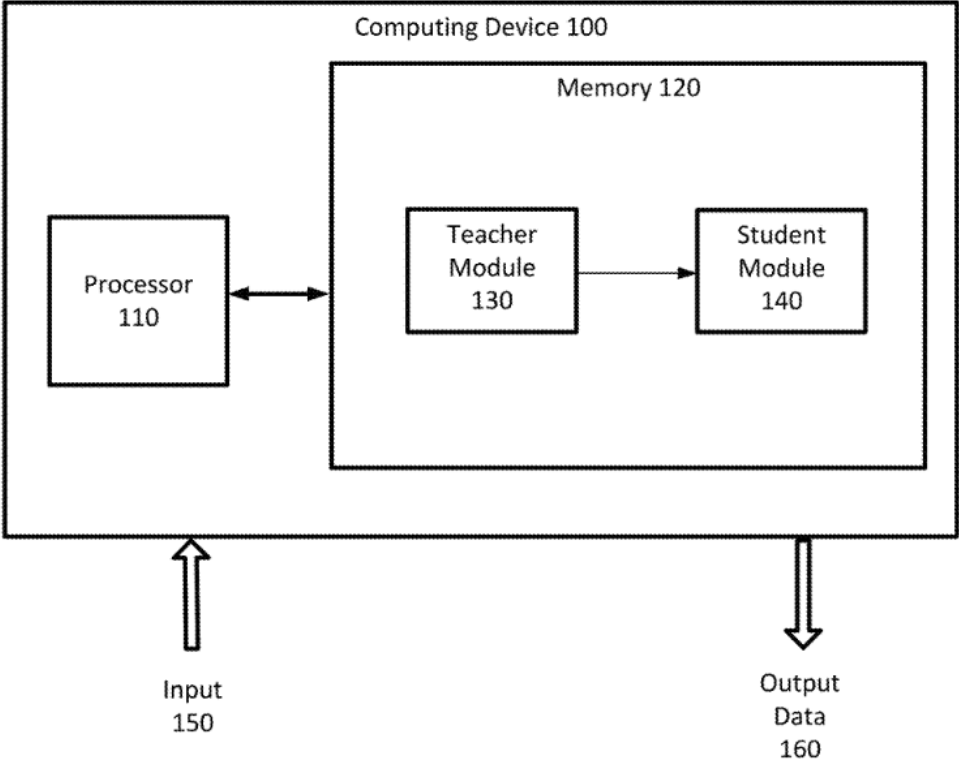
- The proposed unified transformer vision and dialogue BERT model has several advantages over traditional visual dialogue systems.
- The model is able to jointly encode visual and linguistic information, which allows it to better understand and respond to complex queries.
- The model is also able to fine-tune on a variety of tasks, which makes it more versatile and applicable to a wider range of applications.

Conclusion

- The proposed unified transformer vision and dialogue BERT model is a powerful new tool for answering questions about images and dialogue.
- The model has several advantages over traditional visual dialogue systems, and it is well-suited for a variety of applications.
- Future work could focus on further improving the model's accuracy and efficiency, as well as expanding its capabilities to new tasks.

Article #22 Notes: Multi-task knowledge distillation for language model

Source Title	Multi-task knowledge distillation for language model
Source citation (APA Format)	Liu, L., & Xiong, C. (2023, April 4). Multi-task knowledge distillation for language model. Google Patents. https://patents.google.com/patent/US11620515B2/en
Original URL	https://patents.google.com/patent/US11620515B2/en
Source type	Patent
Keywords	Multi-task Knowledge Distillation, Language Model, Shared Layers, Task Layers, Teacher Model
#Tags	#nlp, #dl, #knowledgedistillation, #ml, #multitasklearning
Summary of key points + notes (include methodology)	<p>This patent describes a method for training language models using multi-task knowledge distillation. The method involves training a larger teacher model on a large corpus of text data, and then training a smaller student model on a smaller corpus of data. The student model is trained to mimic the predictions of the teacher model, and is also trained on the task-specific data for the specific tasks that the student model is intended to perform. This method allows for the transfer of knowledge from the teacher model to the student model, which can improve the performance of the student model on the task-specific data. The patent also describes a specific architecture for the teacher and student models. The teacher model has a number of shared layers that are responsible for extracting common features from the input data. The student model also has a number of shared layers, but the student model also has a number of task-specific layers that are responsible for performing the specific tasks that the student model is intended to perform. The shared layers are initialized with the weights of the corresponding shared layers in the teacher model. The patent also describes a method for training the student model. The student model is first trained on the task-specific data using a standard backpropagation algorithm. The student model is then fine-tuned on the task-specific data using a method that takes into account the predictions of the teacher model. The fine-tuning process involves adjusting the weights of the student model's task-specific layers so that the student model's predictions more closely match the predictions of the teacher model. The patent claims that the method described in the patent can improve the performance of language models on a variety of tasks. The patent also claims that the method can be used to train language models that are smaller and more efficient than traditional language models.</p>

Research Question/Problem/ Need	How can we train language models more efficiently by transferring knowledge from a larger model to a smaller model?
Important Figures	 <p>Diagram of the computing workflow</p>

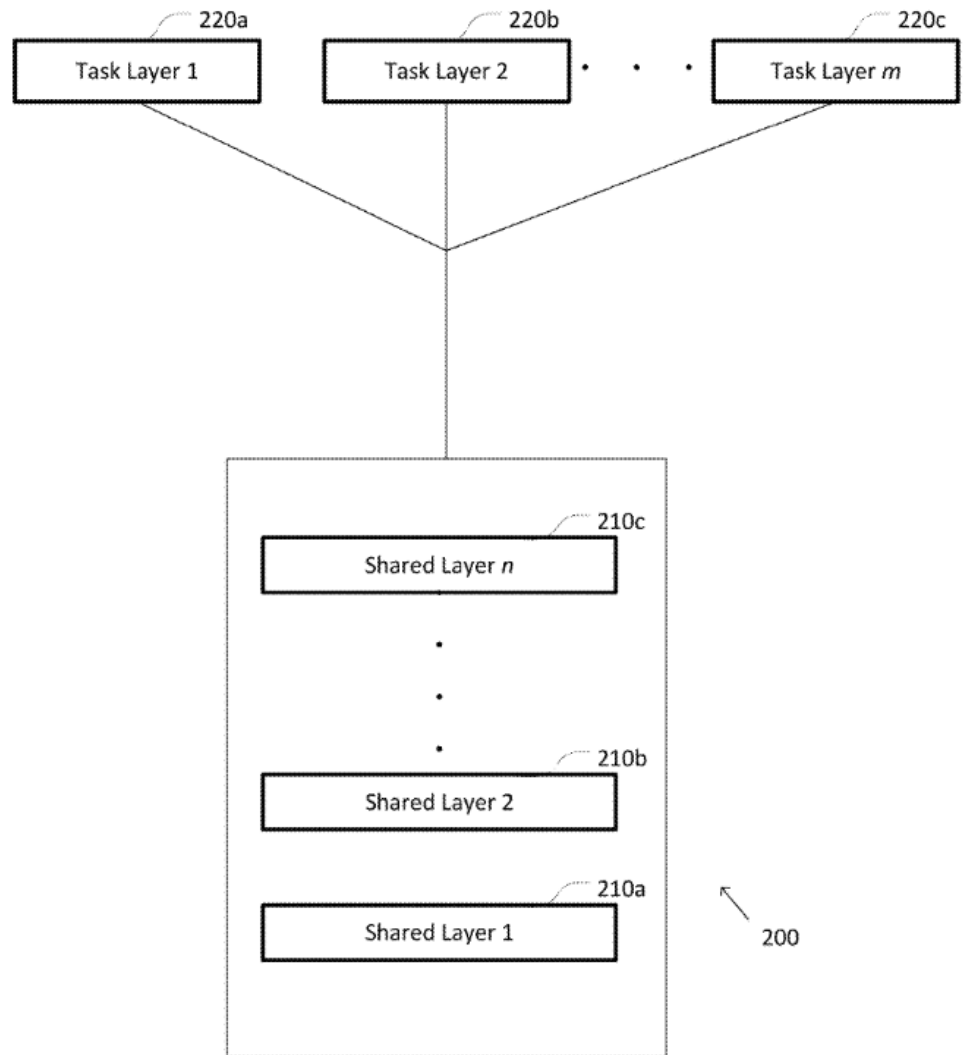
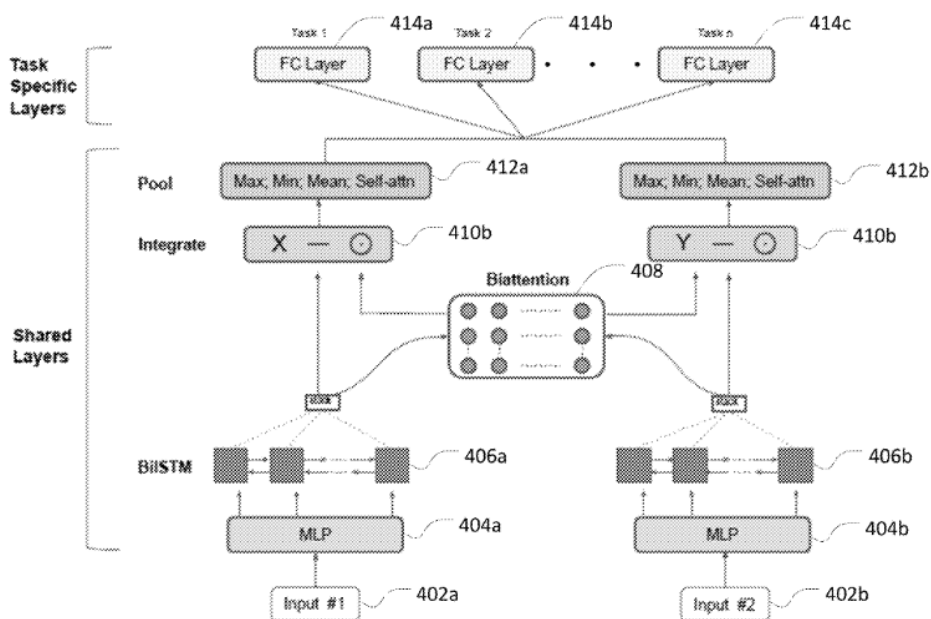


Diagram of the layers



The workflow and inputs, outputs, pooling, and mechanisms of the various layers

VOCAB: (w/definition)

Multi-task learning: Training a model to perform multiple tasks simultaneously, like summarizing text and answering questions about it.

Knowledge distillation: Transferring knowledge from a larger, "teacher" model to a smaller, "student" model, improving the student's performance.

Shared layers: Layers in a neural network that both the teacher and student models use, extracting common features from data.

Task-specific layers: Layers in a neural network specific to the tasks the student model is designed for (e.g., translation, question answering).

Fine-tuning: Further training a model on specific data to improve performance on those tasks.

Backpropagation: An algorithm used to adjust weights in a neural network based on its errors, helping it learn.

Corpus: A large collection of text data used for training language models.

Feature extraction: Identifying and highlighting important characteristics of data.

Parametric model: A model with a limited set of adjustable parameters (weights) that determine its predictions.

	Non-parametric model: A model with more flexible structures that adapt to data automatically, not relying solely on fixed parameters.
Cited references to follow up on	Clark et al., "Bam! Born-Again Multi-Task Networks for Natural Language Understanding," arXiv:1907.04829, pp. 1-7, 2019.
Follow up Questions	<p>The patent mentions the potential for smaller, more efficient language models. Can you think of any specific applications where this could be beneficial, like in chatbots or virtual assistants? How might these models impact our daily lives?</p> <p>While the patent highlights the advantages of multi-task learning, are there any potential drawbacks or limitations to this approach? For example, could focusing on multiple tasks compromise the model's performance on any specific one?</p> <p>The patent represents one approach to training language models. Can you imagine any other innovative techniques or advancements that could be developed in the future? How might this field continue to evolve in the coming years?</p>

Notes (written with the assistance of ChatGPT)

Background:

- What problem is being addressed by the patent?
- What existing solutions are there, and why are they not ideal?
- What are the key terms and concepts introduced in this section?

Summary of the Invention:

- What is the main innovation described in the patent?
- How does it work? What are the key steps or components?
- What are the potential benefits of this invention?

Detailed Description:

- Break down the section into smaller parts (e.g., figures, specific techniques).
- Note down key points for each part, focusing on specific details and technical terms.
- Draw diagrams or flowcharts if helpful to visualize the process.

Claims:

- What are the legal claims protecting the invention?
- What features or aspects of the invention are considered unique and novel?
- How do the claims relate to the technical details described earlier?

Conclusion:

- What are the main takeaways from this patent?
- How does it contribute to the field of language models?
- What are the potential future applications or directions based on this work?

Article #23 Notes: Adversarial pretraining of machine learning models

Source Title	Adversarial pretraining of machine learning models
Source citation (APA Format)	Liu, X., Cheng, H., Wang, Y., Gao, J., Chen, W., He, P., & Poon, H. (2023, October 31). Adversarial pretraining of machine learning models. Google Patents. https://patents.google.com/patent/US11803758B2/en
Original URL	https://patents.google.com/patent/US11803758B2/en
Source type	Patent
Keywords	Machine learning models, Pretraining, Natural language processing, Transformer encoder, Self-supervised learning
#Tags	#mlmodels, #nlp, #transformers, #adversarialtraining
Summary of key points + notes (include methodology)	This article dives into a new technique called adversarial pretraining, designed to make machine learning models better at understanding language. It works by adding a bit of "fuzz" to the initial understanding of training examples, then using that to train the model itself. Think of it like giving a language learner scrambled sentences to practice with, making them better at deciphering real language later. This approach, called noise-adjusted first representations, has been shown to boost the model's performance on various language tasks, like figuring out what words go together or grasping the meaning of a sentence. It even outperforms other training methods in some cases! Overall, adversarial pretraining with noise-adjusted representations seems like a promising way to train language models, potentially leading to more accurate and versatile language processing tools in the future.
Research Question/Problem/Need	Can adding "fuzz" to training examples improve the performance of machine learning models for natural language processing tasks?

Important Figures

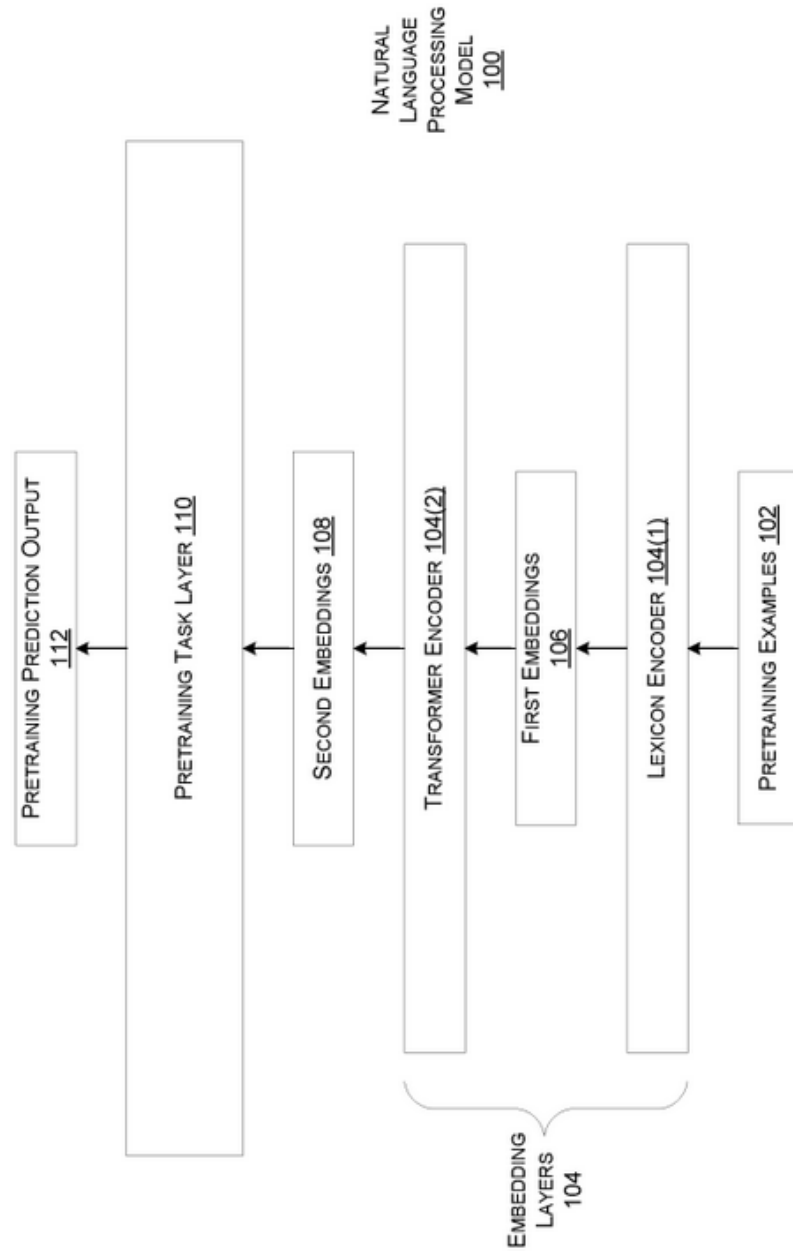
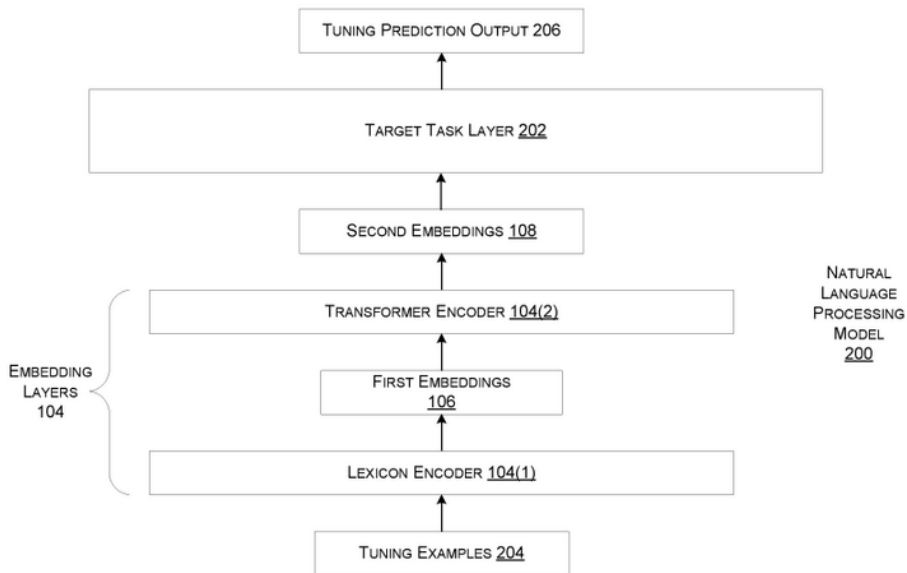


FIG. 1

This figure shows the architecture of the proposed adversarial pretraining method



An encoding/embedding/output workflow

Model	SQuAD v1.1/v2.0		MNLI
	F1/EM	F1/EM	m/mm Acc
BERT <small>BASE</small>	88.5/81.0	76.5/72.9	84.5/84.4
BERT+ <small>BASE</small>	89.6/82.4	77.8/74.0	85.0/84.8
ALUM <small>BERT-BASE</small>	90.8/83.7	80.2/76.6	85.8/86.1

This figure shows the results of the experiments on the MNLI benchmark

VOCAB: (w/definition)

Adversarial Pretraining: A method of training machine learning models by adding noise to the training data and then using a self-supervised learning process to learn to predict the original data from the noisy data.

Machine Learning Models: Models that are trained on data to learn to perform a specific task, such as recognizing objects in images or generating text.

Pretraining: A process of training a machine learning model on a large corpus of data before using it for a specific task.

Natural Language Processing (NLP): A field of computer science that deals with the interaction between computers and human language.

Transformer Encoder: A neural network architecture that is commonly used for NLP tasks.

Noise-Adjusted First Representations: The representations of the training data that

	<p>are obtained after adding noise to the original representations.</p> <p>Self-Supervised Learning: A type of machine learning that learns to perform a task without the need for labeled data.</p> <p>Mapping Layers: The layers of a neural network that are responsible for mapping the input data to a higher-dimensional representation.</p> <p>Pretraining Examples: The individual pieces of data that are used to train a pretraining model.</p> <p>Neural Language Models: A type of machine learning model that is specifically designed for NLP tasks.</p>
Cited references to follow up on	<p>Cer, et al., "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-Lingual Focused Evaluation", In Journal of Computing Research Repository, Jul. 31, 2017, 14 Pages.</p>
Follow up Questions	<p>The article focuses on improving model performance on benchmark tasks. Can you think of some specific real-world scenarios where this type of adversarial pretraining might be used to improve language understanding in practical applications, like chatbots or virtual assistants?</p> <p>While the article shows impressive results, are there any potential limitations or risks associated with using adversarial pretraining? For example, could the "fuzz" introduced during training lead to unintended biases or errors in the model's output?</p> <p>The research presented here represents a significant step forward in NLP. What are some promising research directions or open questions that this work could lead to? Could this technique be applied to other types of data beyond text, or could it be used to improve other areas of machine learning besides natural language processing?</p>