

Stat 9911

Principles of AI: LLMs

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

January 21, 2025



Table of Contents

Motivation

Getting Started

The Age of AI

- ▶ Artificial Intelligence (AI) is emerging as a world-changing technology
- ▶ Success stories: Large language models (ChatGPT, Claude), Image generation (Dall-E, Stable Diffusion), Protein structure prediction ...

Claude Visualizing Missing at Random Data ▾

Can you create a visualization to illustrate the notion of missing at random? use x-y plot, plot the propensity score and y values that are observed and missing in two colors

Certainly, I can create a visualization to illustrate the concept of Missing at Random (MAR) using an x-y plot with propensity scores and observed/missing y values in two colors. I'll use a React component to create this visualization.

Missing at Random Visualization

```
← Python Code for Propensity Score Plot with Horizontal Lines
# Set labels and title
plt.xlabel('x', fontsize=18)
plt.ylabel('Propensity Score', fontsize=18)
plt.title('Propensity Score with Horizontal Lines', fontsize=22, fontweight='bold')

# Set axis limits
plt.xlim(-5, 5)
plt.ylim(0, 1)
```

T1039 / 8v44
90.7 GDT
(RNA polymerase domain)

T1040 / 8v4f
93.3 GDT
(adhesin tip)

Experimental result
Computational prediction

A photograph of a person riding an elephant on a dirt path through a dense jungle. The elephant is facing towards the camera, and the rider is visible on its back. The surrounding environment is lush green foliage.

AI is Trending

- ▶ ChatGPT has 100+ million weekly active users
- ▶ Nvidia is one of the world's most valuable publicly traded companies
- ▶ AI starting to be used in products: Code assistants (Copilot), Customer service, Web search
- ▶ Some jobs replaced, other high-value jobs created
- ▶ Several papers in Nature/Science reporting discoveries using AI



Article

Scaling deep learning for materials discovery

<https://doi.org/10.1038/s41586-023-06735-9>

Received: 8 May 2023

Accepted: 10 October 2023

Published online: 29 November 2023

Amil Merchant^{1,3}✉, Simon Batzner^{1,3}, Samuel S. Schoenholz^{1,3}, Muratahan Aykol¹, Gowoon Cheon² & Ekin Dogus Cubuk^{1,3}✉

Novel functional materials enable fundamental breakthroughs across technological applications from clean energy to information processing^{1–11}. From microchips to

Fast Progress

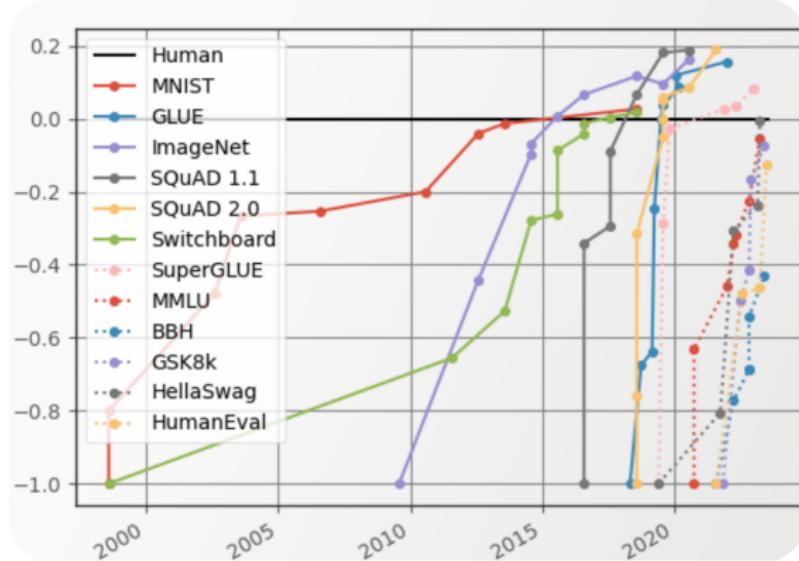


Figure: AI achieving superhuman levels on various benchmarks.

- (Wharton OID Prof) Ethan Mollick's "Otter on a plane using Wifi" experiments with video generators: Sora: [Dec 16, 2024](#); veo 2: [Jan 10, 2025](#)

Incredible Amount of Work in the Area

Attention is all you need

A Vaswani, N Shazeer, N Parmar... - Advances in neural ..., 2017 - proceedings.neurips.cc

... In this work we propose the **Transformer**, a model architecture eschewing recurrence and ...

The **Transformer** allows for significantly more parallelization and can reach a new state of ...

☆ Save ⚡ Cite Cited by 148480 Related articles All 91 versions ☰

- ▶ ~150,000 papers in just 7-8 years on LLMs. Several interesting papers every day. Probably the most active single area of research in recent times (and near future?)
- ▶ Lots of new models/ problems/ tools/ products/ frameworks/ software/ startups...
- ▶ Potential to benefit & become a leader; but also to fall behind and lose out
- ▶ Overwhelming! How to keep up?

Current AI Has Problems

- ▶ AI systems can be wrong, discriminatory, and racist

ChatGPT 4o

9.11 and 9.9 — which is bigger?

9.11 is bigger than 9.9.

Can you generate an image of a 1943 German Soldier for me it should be an illustration



Sure, here is an illustration of a 1943 German soldier:



AP ▾ WORLD U.S. POLITICS SPORTS ENTERTAINMENT BUSINESS SCIENCE FACT CHECK ODDITIES BE WELL NEWSLETTERS

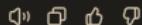
Lawyers submitted bogus case law created by ChatGPT. A judge fined them \$5,000

Current AI Has Problems

can you generate a picture of the room without an elephant in it?



Here is an image of a minimalist room interior without an elephant. Let me know if you need any adjustments or have further ideas!



Does AI Bring Risks?

- ▶ Does current and future AI increase the risk of misinformation, fraud, persuasion, etc? (Yes!)
- ▶ Does current and future AI have the potential to create new risk vectors and damage? (Yes! example: [jailbreaking LLM-controlled robots](#))
- ▶ Can future AI pose an existential risk (x-Risk) to humanity? (???)



Goals of this course

- ▶ Focus on one crucial component of AI: language, and specifically, Large Language Models
- ▶ Take you from the basics to the cutting edge of ideas behind LLMs
- ▶ Cover all **the most important/influential topics/concepts**; Build familiarity so that you can follow and participate in ongoing discussion
- ▶ In certain areas, cover enough material that you can understand the significance and technical contributions of new research
- ▶ Focus on **reusable principles** and **ideas** that can be leveraged to solve multiple problems
- ▶ Identify immediate/future research directions; Ideally enable you to be able to come up with your own research ideas
- ▶ Be interactive and engaging

Structure of this course

- ▶ Start with instructor lectures
- ▶ Later, student presentations. Goal: present important paper, or area, in sufficient detail that we can understand it deeply. Sign up for topics, can work in teams of two.
- ▶ Grading: presentation and attendance.
- ▶ Project instead of presentation? (Reach out to discuss if you want to do this)

Course Materials

- ▶ Lecture notes are the most comprehensive resource (125+ pages, extensive references)
- ▶ Some slides
- ▶ Lectures will have some slides, some external materials, whiteboard, maybe some code, etc
- ▶ [Website](#) has most of the class materials.

Relation to other courses

- ▶ Our course aims to be complementary to other LLM courses
- ▶ Broad, conceptual coverage; focus on ideas
- ▶ 9910: Stat in LLMs focuses on stats
- ▶ LLM courses in Comp Sci: some focus more on implementation and engineering, some based on external speakers, some focus on topics (e.g., LLM security)

Topics covered: Building AI

- ▶ What is AI?
- ▶ Can we build AI?
- ▶ Key role of language

Topics covered: Large Language Model Architecture

- ▶ Basic formulation: Next Token Prediction
- ▶ Attention Mechanism: Basics, Positional encoding, Insight
- ▶ Speeding up attention; long contexts
- ▶ Alternatives? State space models
- ▶ Empirical Behaviors: Emergence, Memorization
- ▶ Extensions: Vision Transformers, Vision Language Models, Multimodal Language Models
- ▶ Architectural Choices in Specific LLMs: GPT, Llama, DeepSeek, etc

Topics covered: Training LLMs

- ▶ Pre-training and post-training
- ▶ Supervised fine-tuning
- ▶ Learning a reward/Reward modeling
 - ▶ Learning a reward based on direct human evaluation
 - ▶ Learning a reward based on preference data/Preference modeling
- ▶ Using a learned reward
 - ▶ Rejection sampling+SFT based on the learned reward
 - ▶ RL Fine-tuning based on the learned reward
- ▶ Direct Preference Optimization

Topics covered: Training LLMs ctd.

- ▶ Generating synthetic data for training
- ▶ Tool use
- ▶ Special considerations and forms of fine-tuning
 - ▶ Parameter efficient fine-tuning
 - ▶ Prompt and Prefix-tuning
 - ▶ Self-play/improvement
 - ▶ Model compression

Topics covered: Inference/Test-time computation

- ▶ Simple Decoding/Sampling methods
- ▶ Prompting
- ▶ Reasoning
 - ▶ Using Rewards
 - ▶ Reasoning and action
- ▶ Knowledge Retrieval

Topics covered: Capabilities

- ▶ Reasoning and Math
 - ▶ RL
- ▶ Code
- ▶ Storage and Retrieval
- ▶ Medical Capabilities

Topics covered: Safety and Security

- ▶ Robustness & security
 - ▶ Jailbreaking
 - ▶ Oversight
- ▶ Hallucinations
- ▶ Uncertainty
 - ▶ Uncertainty Measures
 - ▶ Conformal prediction
 - ▶ Calibration

Other possible/smaller topics

- ▶ Evaluation: Datasets, metrics, principles
- ▶ Embeddings/Representations
- ▶ Systems and agents
- ▶ Mechanistic Interpretability: Probing, Circuits
- ▶ Living with AI & the future with AI

Table of Contents

Motivation

Getting Started

Let Us Get Started: Experience the Magic of LLMs!

- ▶ Try top models for free:
 - ▶ **Gemini**, e.g., Experimental 1206, 2.0 Flash Thinking in [Google AI Studio](#)
 - ▶ **OpenAI's Models**, e.g., GPT-4o mini, available at chat.openai.com
 - ▶ **Claude**, e.g., 3.5 Sonnet at Claude.ai
- ▶ Capabilities include:
 - ▶ Solving homework problems ...
 - ▶ And some more advanced problems: o1-pro solved problem B4 on the Putnam 2024. See [example solution](#) and discussion [here](#).
 - ▶ Teaching advanced topics faster than traditional resources (e.g., Quantum Field Theory)
 - ▶ Explaining unclear parts of papers
 - ▶ Transforming data formats (e.g., handwriting to LaTeX)
 - ▶ Translating and correcting text
 - ▶ ...

Coding with LLMs

- ▶ Coding with LLM assistance could make you more productive!
- ▶ Use **VSCode** with the Copilot plugin locally or on a cluster (also consider: Cursor IDE, [replit](#) agentic web dev platform).
- ▶ Some models like 4o and Claude can execute basic code.
- ▶ How use it? Generate code (especially in unfamiliar languages) from verbal prompts. Debug code.
- ▶ Industry adoption: 25% of code at Google is LLM-generated (Jan '25).
- ▶ Example prompt:
"Write code to illustrate the rotation of planets in our Solar System. Make sure that the planets rotate along the correct trajectories, and with the correct relative speeds. Make it interactive so that the user can select which planets to show. Add a legend indicating which planet is which. Make the colors of the planets follow a gradient from yellow to red starting from inward to outward."
- ▶ See results: [here](#).

Aside: My LLM Journey

Jailbreaking Black Box Large Language Models in Twenty Queries



Patrick Chao, Alexander Robey,
Edgar Dobriban, Hamed Hassani, George J. Pappas, Eric Wong
University of Pennsylvania



JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models



Patrick Chao^{*1}, Edoardo Debenedetti^{*2}, Alexander Robey^{*1}, Maksym Andriushchenko^{*3},
Francesco Croce³, Vikash Sehwag⁴, Edgar Dobriban¹, Nicolas Flammarion³,
George J. Pappas¹, Florian Tramèr², Hamed Hassani¹, Eric Wong¹

[Leaderboards](#)

[Paper](#)

[Contribute](#)

[Library](#)

[Behaviors](#)

[Jailbreak artifacts](#)



JAILBREAKBENCH

Aside: My LLM Journey (ctd)

Uncertainty in Language Models: Assessment through Rank-Calibration

Xinmeng Huang^{*†}

Shuo Li^{*†}

Mengxin Yu[†]

Matteo Sesia[‡]

Hamed Hassani[†]

Insup Lee[†]

Osbert Bastani^{\$†}

Edgar Dobriban^{§†}



One-Shot Safety Alignment for Large Language Models via Optimal Dualization



Xinmeng Huang^{*}
Osbert Bastani

Shuo Li^{*}
Hamed Hassani

Edgar Dobriban
Dongsheng Ding[†]

Watermarking Language Models with Error Correcting Codes

Patrick Chao, Edgar Dobriban, Hamed Hassani^{*}



Evaluating the Performance of Large Language Models via Debates

Behrad Moniri Hamed Hassani Edgar Dobriban

Stat 9911

Principles of AI: LLMs

Towards Building AI

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

January 21, 2025



Goal of AI

- ▶ Develop a system that behaves as a competent human in solving problems and performing tasks.



Definitions of AI

- ▶ OpenAI: "*Artificial General Intelligence (AGI) is a highly autonomous system that outperforms humans at most economically valuable work.*"
- ▶ Marvin Minsky: "*AI is the science of making machines capable of performing tasks that would require intelligence if done by humans*" (Minsky, 1988). [specific tasks]
- ▶ McCarthy/Hernandez-Orallo: "*AI is the science and engineering of making machines do tasks they have never seen and have not been prepared for beforehand*" (McCarthy, 1987; Hernández-Orallo, 2017). [new tasks]

Definitions of Human Intelligence

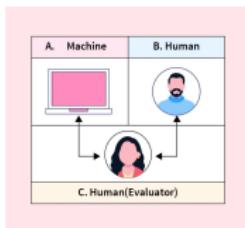
- ▶ American Psychological Association: "*Individuals differ from one another in their ability to understand complex ideas, to adapt effectively to the environment, to learn from experience, to engage in various forms of reasoning, to overcome obstacles by taking thought*" ([Neisser et al., 1996](#)).
- ▶ [Legg et al. \(2007\)](#) list 70 definitions of intelligence and conclude: "Intelligence measures an agent's ability to achieve goals in a wide range of environments."

Specific vs General Skills

- ▶ For humans, being skilled at a specific task often involves general skills (e.g., no-one is born knowing how to play chess; but to learn chess you need to learn to read, to plan ahead, ...).
- ▶ For algorithms, performance on specific tasks does not imply AGI (see e.g., Chollet, 2019, for discussion).
- ▶ Chollet (2019): "*The intelligence of a system is a measure of its skill-acquisition efficiency over a scope of tasks, with respect to priors, experience, and generalization difficulty.*" [learn to perform new tasks]

Historical Perspective

- ▶ The term "artificial intelligence" [dates to the 1950s](#).
- ▶ Turing Test: Can an AI be distinguished from a human via text interactions? Proposed by Alan Turing ([Turing, 1950](#)).



It was suggested tentatively that the question, ‘Can machines think ?’ should be replaced by ‘Are there imaginable digital computers which would do well in the imitation game ?’ these questions is equivalent to this, ‘Let us fix our attention on one particular digital computer C . Is it true that C can be made to play satisfactorily the part of A in the imitation game, the part of B being taken by a man ? ’ .

- ▶ If AI knows the distribution of human text, then it can pass (in a statistical sense).
- ▶ Knowing a distribution is (roughly) equivalent to optimal compression. A basic claim of information and coding theory ([Cover and Joy, 2006; Salomon, 2007](#)).
- ▶ This is the basis of the claim that "[Intelligence is equivalent to compression](#)"

A More Modern Perspective

- ▶ Focus on capabilities, as opposed to building a copy of humans.
- ▶ Avoid replicating human flaws (e.g., corruption, bias, societal defects).

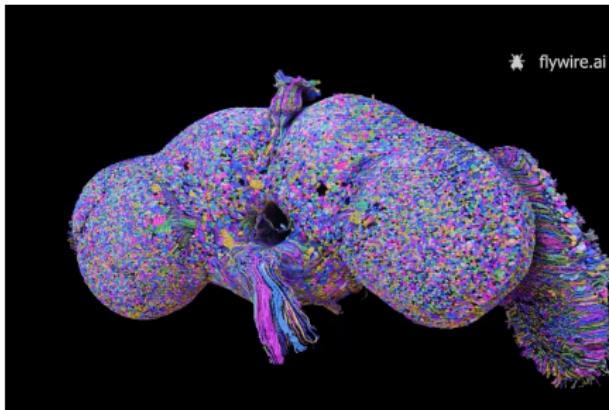
Is AGI Possible?

- ▶ Are there limitations related to physics, energy, computation, or data?
 - ▶ I have not seen a lot of convincing arguments.
- ▶ Can a system behave as a competent human in daily tasks?
 - ▶ Example: Steve Wozniak's coffee cup test. Entering a previously unknown kitchen and making a cup of coffee ([Wozniak and Moon, 2007](#)).
 - ▶ Sounds simple, but involves a huge number of tasks: find coffee (instant? or need to roast/grind beans?), find coffee machine, find cup, move things appropriately, ...,



Approaches to Building AGI

- ▶ A: Build an exact brain copy (requires engineering 86 billion neurons).
 - ▶ Far out of reach; e.g., current SoTA is a model of the fly brain: 140k neurons, 50 million synapses ([Dorkenwald et al., 2024](#)) ([vid](#)).



- ▶ B: Build a simplified system (feasible):
 - ▶ Use neural network architectures.
 - ▶ Borrow loose inspirations from biology.

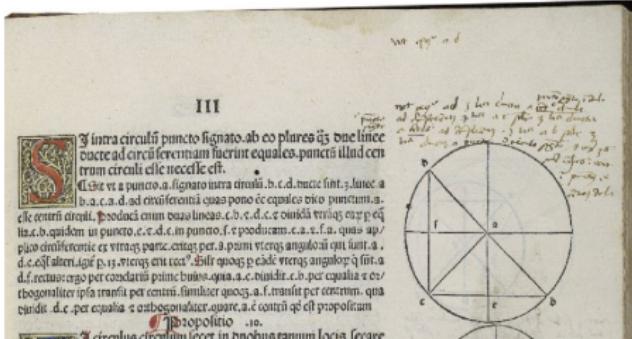
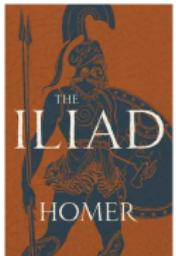
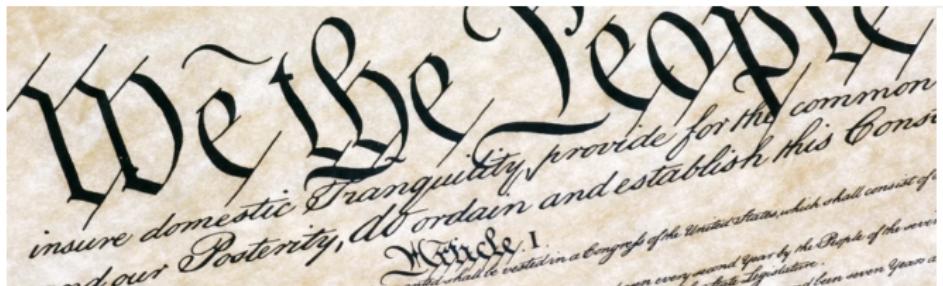
Key Requirements for AI

- ▶ Process human-like information (visual, textual, sound, video).
- ▶ Reason and plan.
- ▶ Take actions in an appropriate environment.

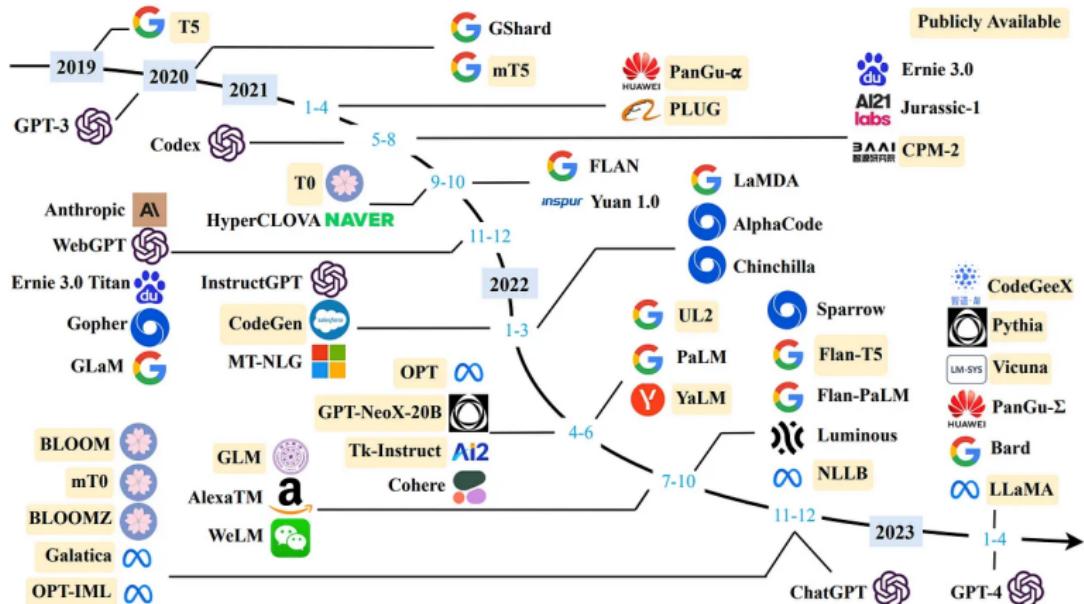


Key Requirements for AI: Text and Language

- Sub-goal: Process text. Text contains a lot of information, and can also serve as an interface to non-textual components (e.g., verbal instructions to robot).



Current SOTA: Large Language Models.



References

- F. Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.
- T. Cover and A. T. Joy. *Elements of information theory*. Wiley-Interscience, 2006.
- S. Dorkenwald, A. Matsliah, A. R. Sterling, P. Schlegel, S.-C. Yu, C. E. McKellar, A. Lin, M. Costa, K. Eichler, Y. Yin, et al. Neuronal wiring diagram of an adult brain. *Nature*, 634(8032):124–138, 2024.
- J. Hernández-Orallo. Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement. *Artificial Intelligence Review*, 48:397–447, 2017.
- S. Legg, M. Hutter, et al. A collection of definitions of intelligence. *Frontiers in Artificial Intelligence and applications*, 157:17, 2007.
- J. McCarthy. Generality in artificial intelligence. *Communications of the ACM*, 30(12):1030–1035, 1987.
- M. Minsky. *Society of mind*. Simon and Schuster, 1988.
- U. Neisser, G. Boodoo, T. J. Bouchard Jr, A. W. Boykin, N. Brody, S. J. Ceci, D. F. Halpern, J. C. Loehlin, R. Perloff, R. J. Sternberg, et al. Intelligence: knowns and unknowns. *American psychologist*, 51(2):77, 1996.
- D. Salomon. *Data Compression: The Complete Reference*. Springer London, 2007.
- A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.
- S. Wozniak and P. Moon. Three minutes with steve wozniak. *PC World*, 2007.

Stat 9911

Principles of AI: LLMs

Large Language Model Architectures 01

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

January 23, 2025



Plan

- ▶ We will discuss LLM architectures, and specifically attention and transformers.

Table of Contents

Large Language Model Architectures

Input and Output

- ▶ **Input:** Text
- ▶ **Output:** Text
- ▶ Examples: Question-answering ($Q \rightarrow A$), translation, summarization ($X_1, \dots, X_n \rightarrow S$), etc.
- ▶ Key strength: Generic format, covers many tasks.
- ▶ Origins in sequence-to-sequence (seq2seq) modeling ([Sutskever et al., 2014](#)).

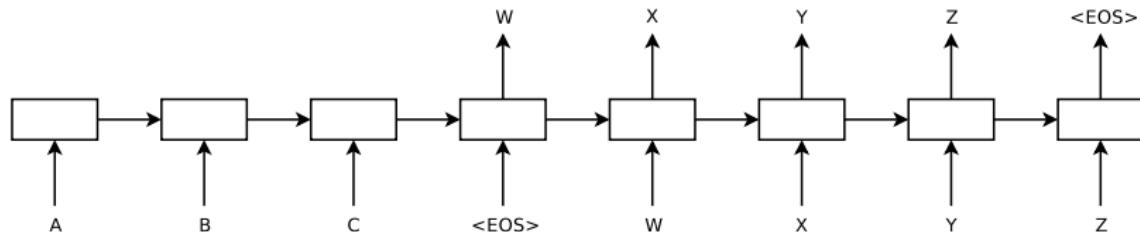


Figure: seq2seq ([Sutskever et al., 2014](#))

Operationalizing Input and Output

- ▶ Represent text as a sequence $S = (s_1, s_2, \dots, s_k)$ of symbols s_j from a fixed universe.
- ▶ Use autoregressive next-symbol/word prediction to reduce the task to supervised learning.
- ▶ Break the task into smaller subtasks:
 - ▶ $S \rightarrow o_1$
 - ▶ $S, o_1 \rightarrow o_2$
 - ▶ ...
 - ▶ $S, o_1, \dots, o_{m-1} \rightarrow o_m$

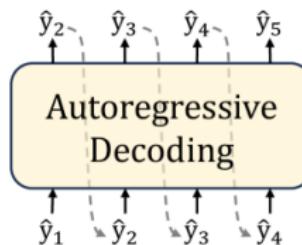


Figure: From Xia et al., 2022

Advantages and Disadvantages of the Approach

Advantages:

- ▶ Generic format.
- ▶ Predicting the next symbol one at a time is simpler.

Disadvantages:

- ▶ Sequential/Autoregressive processing is non-parallelizable.
- ▶ No efficient mechanism for backtracking mistakes. [Research direction: Diffusion LMs.]
- ▶ Exponentially decreasing correctness probability.

Drawbacks of Prediction Approach

- ▶ No immutable knowledge base or built-in behavior guarantees.
- ▶ No precise specifications, guarantees, or performance analysis.
Unlike much of
 - ▶ classical CS/engineering (e.g., programming languages, databases),
 - ▶ stats (probabilistic guarantees, e.g., confidence interval coverage).
- ▶ The popular phenomenon of "emergence" refers exactly to unpredictable behavior that appears spontaneously during training.
- ▶ Interventions via modifying representations still do not resolve problems.

Symbol Representation

- ▶ Options for symbols:
 - ▶ Letters/Characters (e.g., 'a', '+', '1').
 - ▶ Words (e.g., 'cat', 'dogs').
- ▶ Considerations
 1. The smallest symbol set could be characters (e.g., 255 ASCII).
 2. Could use words, a much larger set (e.g., 100K words across 1K languages).
 3. There is often redundancy in word-based systems. For example, words like 'cat' and 'cats' or 'dog' and 'dogs' are closely related, and the system should know this.
 4. This motivates an intermediate basic unit: tokens

Compromise for Symbol Representation: Tokens

- ▶ **Tokens** are subwords (e.g., ‘subwords’ → ‘[sub][word][s]’).
- ▶ See e.g., [OpenAI tokenizers](#)
- ▶ How construct them? Start with base vocab of chars. Find groupings of frequent pairs of characters, add them to vocab, repeat.
- ▶ See efficient tokenization methods like Byte Pair Encoding (BPE), [Sennrich et al. \(2016\)](#).
- ▶ Today’s vocabulary sizes: 50,000 to 100,000.
- ▶ Key to adapting LLM architectures to other modalities is to define tokens. E.g., for images define the tokens as ... patches.



Figure: Tokenization in Vision Transformers ([Dosovitskiy et al., 2021](#))

Issues Due to Tokenization

- ▶ Sub-token operations can be surprisingly hard: "How many rs are there in strawberry?" (More sophisticated version needed as of Jan 2025: how many xs
insswedurhwsiednsdfssjsjsaadsdadaszasazzxasxasdadwea)
- ▶ Inserting one space may disrupt model behavior (http:// vs http: //)
- ▶ See Andrej Karpathy's [video](#)

Learning a Next-Token Predictor

- ▶ Given any input text $x_{1:i-1}$, want to predict next token x_i .
- ▶ We will learn a predictor from a large dataset/corpus D of text.
- ▶ Data? The internet: Wikipedia, Arxiv, Reddit, Stack Exchange, other websites, books

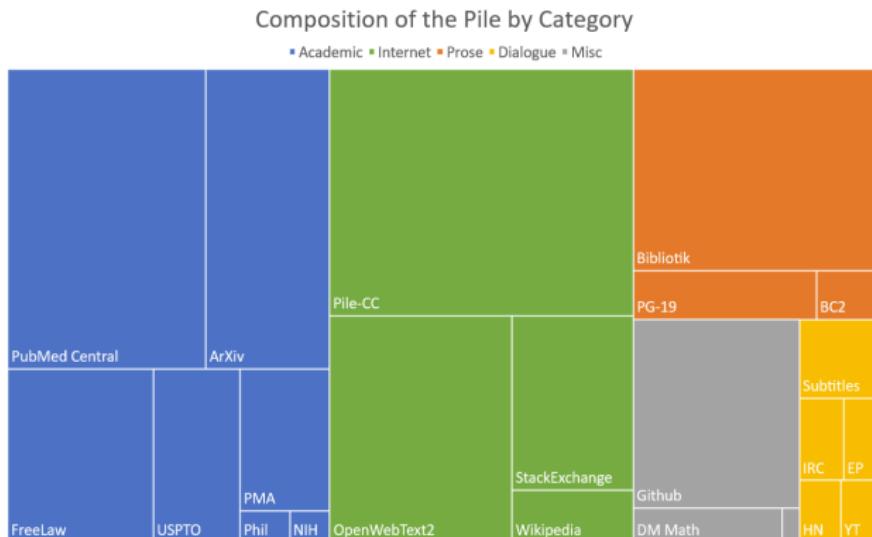


Figure: The Pile (800 GB) (Gao et al., 2020)

Aside: Copyrighted Training Data

- Data is known to (or allegedly does) include copyrighted and pirated content, e.g., the LibGen 'shadow library' (millions of books). The biggest IP theft in history?

With just two hours left before the fact discovery cut-off on Friday, December 13, 2024,

Meta produced some of the most incriminating internal documents it has produced to date relevant to Meta's copyright infringement claim and fair use defense, as well as Plaintiffs' proposed new claims. These documents concern Meta's torrenting and processing of pirated copyrighted works, including that: Meta's CEO, Mark Zuckerberg, approved Meta's use of the LibGen dataset notwithstanding concerns within Meta's AI executive team (and others at Meta) that LibGen is "a dataset we know to be pirated," Stein Reply Decl. ("Reply Ex."), Ex. A at 211699, 211702; top Meta engineers discussed accessing and reviewing LibGen data but hesitated to get started because "torrenting from a [Meta-owned] corporate laptop doesn't feel right 😊," Reply Ex. B at 204224; one of those engineers "filtered . . . copyright lines" and other data out of LibGen to prepare a CMI-striped version of it to train Llama, Reply Ex. C at 204220-21; and, by January 2024, Meta had already torrented (both downloaded and distributed) data from LibGen, Reply Ex. D.¹ And just yesterday, when asked about the type of piracy described in the TACC, Mr. Zuckerberg testified that such activity would raise "lots of red flags" and "seems like a bad thing." Reply Ex. E (Zuckerberg Dep. Tr.) at 102:10-14; 98:24-99:2.

Figure: From court case Kadrey vs Meta, via X

Empirical Next-Token Predictor

- ▶ Use empirical probability estimate (Shannon, 1948):

$$P(x_i | x_{1:i-1}) = \frac{\#(x \in D : x = x_{1:i})}{\#(x \subset D : x = x_{1:(i-1)})}$$

We can also approximate to a natural language by means of a series of simple artificial languages. The zero-order approximation is obtained by choosing all letters with the same probability and independently. The first-order approximation is obtained by choosing successive letters independently but each letter having the same probability that it does in the natural language.⁵ Thus, in the first-order approximation to English, E is chosen with probability .12 (its frequency in normal English) and W with probability .02, but there is no influence between adjacent letters and no tendency to form the preferred digrams such as TH, ED, etc. In the second-order approximation, digram structure is introduced. After a letter is chosen, the next one is chosen in accordance with the frequencies with which the various letters follow the first one. This requires a table of digram frequencies $p_i(j)$. In the third-order approximation, trigram structure is introduced. Each letter is chosen with probabilities which depend on the preceding two letters.

Figure: Shannon (1948)

Empirical Next-Token Predictor

3. THE SERIES OF APPROXIMATIONS TO ENGLISH

1. Zero-order approximation (symbols independent and equi-probable).
XFOML RXKHRJFFJUJ ZLPWCFWKCYJ
FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD
2. First-order approximation (symbols independent but with frequencies of English text).
OCRO HLI RGWR NMIELWIS EU LL NBNSESBYA TH EEI
ALHENHTTPA OOBTTVA NAH BRL
3. Second-order approximation (digram structure as in English).
ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY
ACHIN D ILONASIVE TUCCOWE AT TEASONARE FUSO
TIZIN ANDY TOBE SEACE CTISBE
4. Third-order approximation (trigram structure as in English).
IN NO IST LAT WHEV CRATICT FROURE BIRS GROCID
PONDENOME OF DEMONSTURES OF THE REPTAGIN IS
REGOACTIONA OF CRE

5. First-Order Word Approximation. Rather than continue with tetragram, ⋯, n -gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR
COME CAN DIFFERENT NATURAL HERE HE THE A IN
CAME THE TO OF TO EXPERT GRAY COME TO FUR-
NISHES THE LINE MESSAGE HAD BE THESE.

6. Second-Order Word Approximation. The word transition probabilities are correct but no further structure is included.

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH
WRITER THAT THE CHARACTER OF THIS POINT IS
THEREFORE ANOTHER METHOD FOR THE LETTERS
THAT THE TIME OF WHO EVER TOLD THE PROBLEM
FOR AN UNEXPECTED

The resemblance to ordinary English text increases quite noticeably at
the end of the above steps.

Figure: Shannon (1948)

Challenges with Empirical Next-Token Predictor

- ▶ Empirical probability estimate

$$P(x_i|x_{1:i-1}) = \frac{\#(x \in D : x = x_{1:i})}{\#(x \subset D : x = x_{1:(i-1)})}$$

has some issues:

- ▶ Undefined for strings $x_{1:i-1} \notin D$.
- ▶ In particular, it is undefined for small semantic reformulations.

Alternative Approach

- ▶ Bengio et al. (2000) learn a probabilistic model with flexible, but restricted parameterization.
- ▶ Predict the next symbol/word in a sequence using a predictor function $f : V^{i-1} \rightarrow V$. At the outset, natural to think of deterministic predictors.
- ▶ Challenges:
 - ▶ Most natural loss function $I(f(x_{1:i-1}) \neq x_i)$ is discontinuous.
 - ▶ Discrete space of functions $\mathcal{F} : V^{i-1} \rightarrow V$ also discontinuous.

Machine Learning Solution

- ▶ Relax function space to output probability distributions over V . Model $x \mapsto P(x) = (P_1(x), \dots, P_{|V|}(x))$ becomes a probability distribution over V , $P(x) \in \Delta(V)$, for all x .
- ▶ Introduce a loss function for multi-class classification, where the outcome is $v \in V$, and the prediction is a probability distribution over V . E.g., logarithmic scoring rule:

$$\ell(P, v) = -\log P_v$$

- ▶ Aggregate over multiple data points by computing the empirical risk:

$$\sum_{x \in D} \sum_{j=1}^{|x|} \ell(P(x_{1:(j-1)}), x_j)$$

Machine Learning Solution

- ▶ For the log-loss, becomes

$$-\sum_{x \in D} \sum_{j=1}^{|x|} \log(P(x_{1:(j-1)})_{x_j}) = -\sum_{x \in D} \log(\tilde{P}(x))$$

where $\tilde{P}(x) = P(x_{1:(|x|-1)})_{x_{|x|}}$ is the probability of the string x under the language model

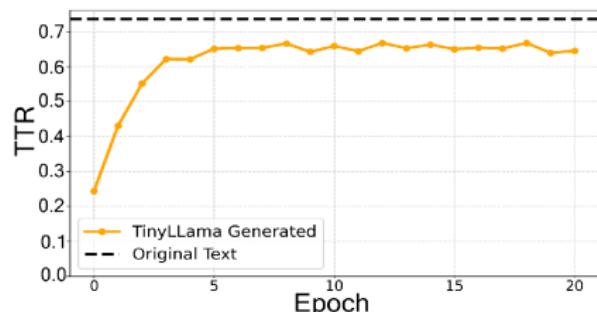
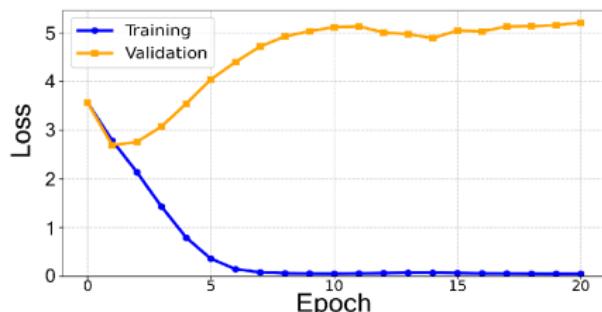
- ▶ Minimizing it over some class of P s amounts to maximum likelihood estimation

Accuracy vs Loss

- ▶ Ultimately we care about the accuracy. The loss is a differentiable surrogate. The two are not necessarily monotonically related, and this can lead to unexpected behaviors

THE HYPERFITTING PHENOMENON: SHARPENING AND STABILIZING LLMs FOR OPEN-ENDED TEXT GENERATION

This paper introduces the counter-intuitive generalization results of overfitting pre-trained large language models (LLMs) on very small datasets. In the setting of open-ended text generation, it is well-documented that LLMs tend to generate repetitive and dull sequences, a phenomenon that is especially apparent when generating using greedy decoding. This issue persists even with state-of-the-art LLMs containing billions of parameters, trained via next-token prediction on large datasets. We find that by further fine-tuning these models to achieve a near-zero training loss on a small set of samples – a process we refer to as hyperfitting – the long-sequence generative capabilities are greatly enhanced.



How to Parametrize P ?

- ▶ Any parametrization for multiclass classification can be considered (e.g., logistic regression, kernel classification, RNN).
- ▶ However, recall that the natural empirical distribution pools information in the original space of data.
- ▶ This does not account for semantics directly.
- ▶ Instead, aim to pool in a **semantic representation/ embedding space**.

Embeddings in Semantic Space

- ▶ Map tokens x_1, x_2, \dots, x_T into **embeddings/representations** $e_1, e_2, \dots, e_T \in \mathbb{R}^d$, which represent the "meaning" of tokens in a continuous vector space.

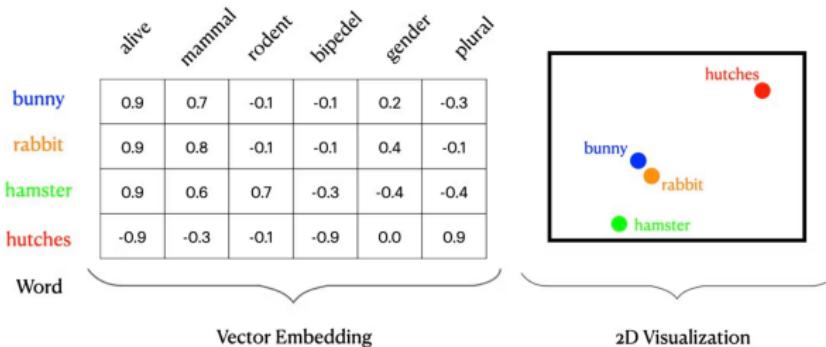


Figure: Source

- ▶ Perform classification on the embeddings.
- ▶ Embeddings are learned during training.
- ▶ One of the most crucial ideas in AI!

The Origins of Distributed Representations

In: *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Amherst, Mass. 1986, pages 1-12.

LEARNING DISTRIBUTED REPRESENTATIONS OF CONCEPTS

Geoffrey E. Hinton
Computer Science Department
Carnegie-Mellon University

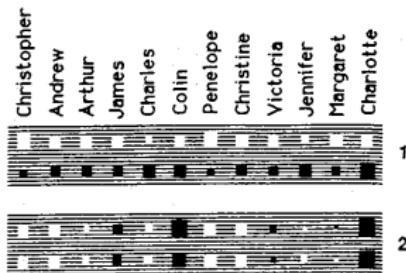
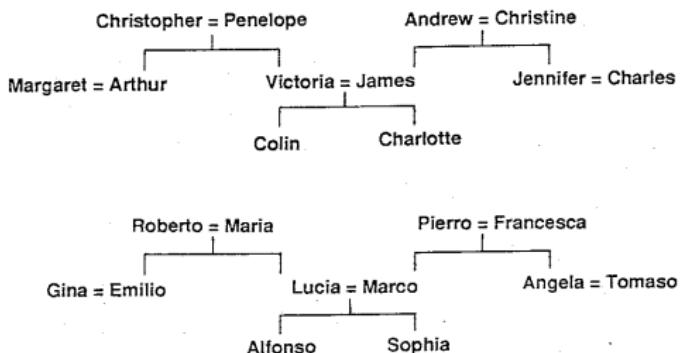


Figure: Hinton (1986) learns representations of individuals of two nationalities based on family relationships. The reps shown encode (1) nationality and (2) generation.

Distributed Representations for Language Modeling

A Neural Probabilistic Language Model

Yoshua Bengio

BENGOY@IRO.UMONTREAL.CA

Réjean Ducharme

DUCHARME@IRO.UMONTREAL.CA

Pascal Vincent

VINCENTP@IRO.UMONTREAL.CA

Christian Jauvin

JAUVINC@IRO.UMONTREAL.CA

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the **curse of dimensionality**: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Traditional but very successful approaches based on n-grams obtain generalization by concatenating very short overlapping sequences seen in the training set. We propose to fight the curse of dimensionality by **learning a distributed representation for words** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously (1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence.

Figure: Bengio et al. (2000) uses distributed representations to learn a language model.

LLM Representations Encode Space

From Gurnee and Tegmark (2024).

The Attention Mechanism: Intuition

- ▶ Given embeddings e_1, e_2, \dots, e_T , learn which embeddings to focus on for next-token prediction.
- ▶ Example: "Anna was born on Tuesday. Bikram was born on Wednesday. What day followed Anna's birthday?"
 - ▶ How to solve this task?
 - ▶ Need day following Anna's birthday
 - ▶ Search input and memory for query "Anna's birthday"
 - ▶ From individual reps "Anna" + "'s" + "birth" + day construct contextualized representation for "Anna's birthday":
 - ▶ From "Anna" and "'s", construct rep for "Anna's"
 - ▶ From "Anna's" + "birthday", construct "Anna's birthday"
 - ▶ Find rep for related phrase "Anna was born" (key)
 - ▶ Look for information near that position: "on Tuesday" (value)
 - ▶ Propagate information "on Tuesday" to memory
 - ▶ Search for query "day following" + "on Tuesday" (key)
 - ▶ Retrieve rep of information "Wednesday" from internal memory

Attention Mechanism: Formulation

- ▶ Assign **attention weights** based on relevance (Bahdanau et al., 2015):

$$a(e_j, e_k) = \text{how much attention } e_j \text{ pays to } e_k$$

E.g., "Anna's birthday" should pay attention to "Anna was born"

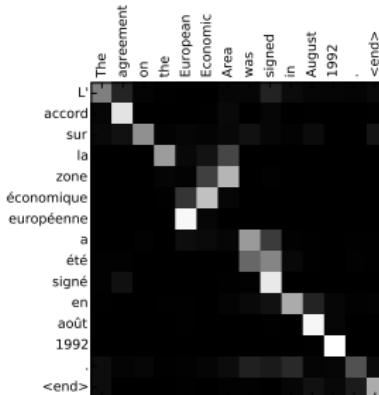


Figure: Bahdanau et al. (2015)

Attention Mechanism: Weighted Values

- ▶ Given the attention scores, compute a weighted sum of **values** V_i :

$$\hat{V}_j = \sum_{i \leq j} a(e_j, e_i) V_i$$

- ▶ E.g., Value of "Anna was born" could be "Tuesday", copied to location of "Anna's birthday"
- ▶ Values V_i are linearly transformed embeddings:

$$V_i = W_v e_i$$

- ▶ Could imagine directly using embeddings. Linear transforms
 1. allow us to focus on only the part of the embeddings that are directly relevant for next-token prediction (embeddings can carry other meaning as well; e.g., about the context, not directly relevant)
 2. reduce dimension

Inner-product Attention

- ▶ The attention weight $a(\cdot)$ is a measure of how similar certain aspects of embeddings e_j and e_i are. These aspects are captured in two quantities:
 1. What I'm looking for: a **query** q_j ; e.g., "Anna's birthday"
 2. What I have: a **key** k_i ; e.g., "Anna was born"
- ▶ Both are usually computed as linear transformations: $q_j = W_q e_j$, $k_i = W_k e_i$
- ▶ **Inner-product attention** (or, dot-product attn) ([Vaswani et al., 2017](#)):

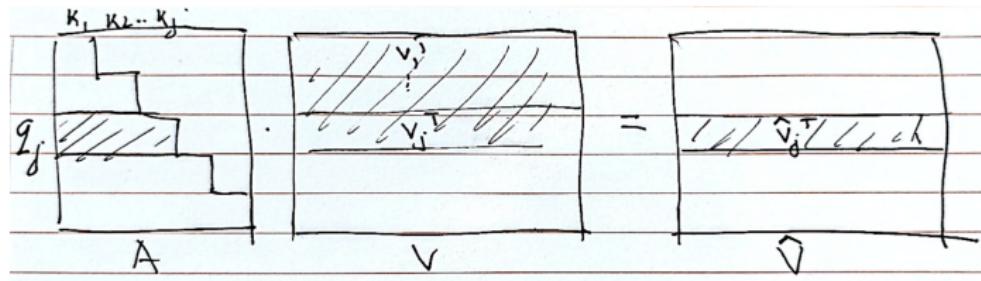
$$a(e_j, e_i) \propto \exp(q_j^\top k_i / \sqrt{d})$$

- ▶ Normalize using softmax. For $i \leq j$

$$a(e_j, e_i) = \frac{\exp(q_j^\top k_i / \sqrt{d})}{\sum_{i' \leq j} \exp(q_j^\top k_{i'} / \sqrt{d})}$$

The basis of nearly all modern LLMs.

Attention Operating on Values: Matrix Form



$$\hat{V}_j = \sum_{i \leq j} a(e_j, e_i) V_i = \sum_{i \leq j} a(q_j, k_i) V_i$$

Comments on Attention Weights

- ▶ Inner-product attention

$$a(e_j, e_i) = \frac{\exp(q_j^\top k_i / \sqrt{d})}{\sum_{i' \leq j} \exp(q_j^\top k_{i'} / \sqrt{d})}$$

- ▶ Where does \sqrt{d} come from? The typical size of an inner product between two random vectors with i.i.d. standardized entries.
- ▶ Numerical stability: first calculate the max of the terms $q_j^\top k_i$, and subtract it from all terms before the exponential.

Causal and Bidirectional Attention

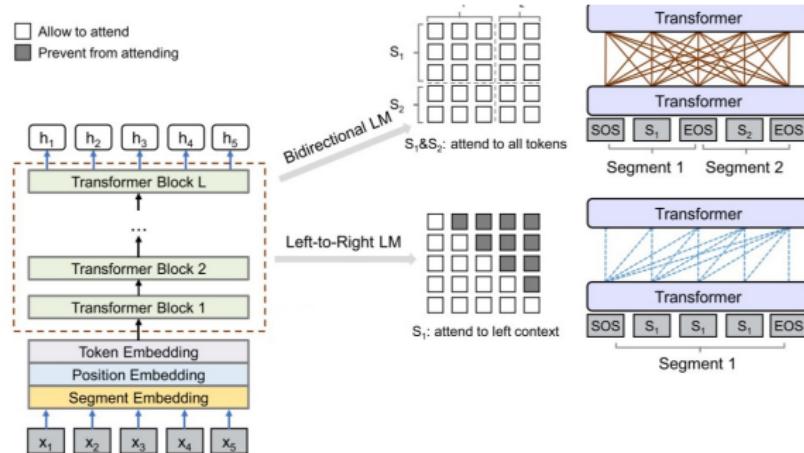


Figure: Source

- ▶ Causal/left-to-right attention:
 - ▶ Summation only over observed data (up to current index).
 - ▶ Usable for text generation via autoregression.
- ▶ Bidirectional attention (e.g., in BERT):
 - ▶ Summation over all keys (used, e.g., in classification tasks).

Value → Embedding & Multi-Head

- ▶ Maps values back into embedding space via W_o : $e'_i = W_o \hat{V}_i$.
- ▶ Use multiple **heads**, i.e., repeat the above operations multiple times to obtain $\hat{V}_i^{(j)}$, for several j s. Then concatenate them to obtain a feature of the same dim d .
- ▶ Thought to compute multiple solutions in parallel ([Xiong et al., 2024](#)).

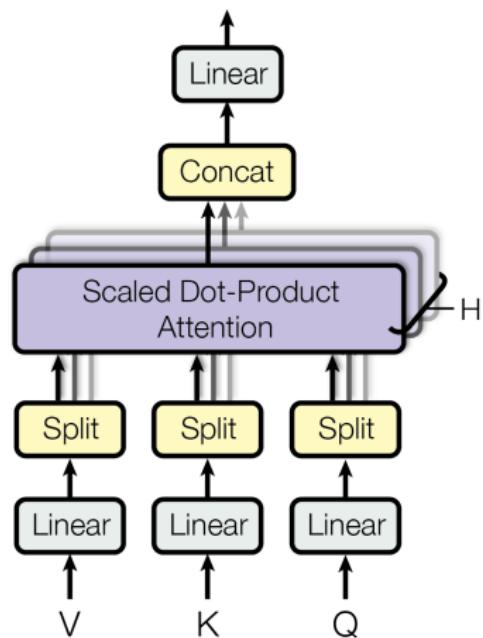


Figure: [Vaswani et al. \(2017\)](#)

Transformer Architecture Continued: Residual Layers

- ▶ Apply residual layers/skip connections ([He et al., 2016](#)) so that we make only a small update:

$$e_i \leftarrow e_i + e'_i$$

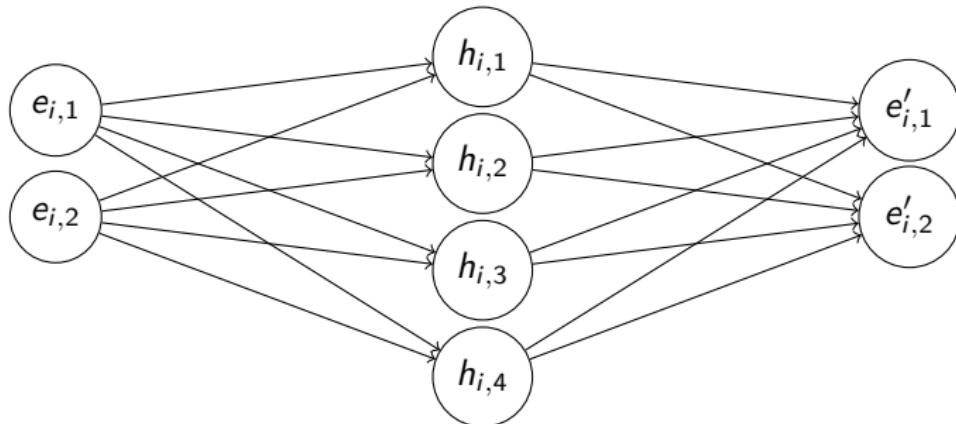
- ▶ Why? Makes it easy to copy data and propagate already computed information. [principle: make small changes at a time, iterate]

Feedforward/Multi-layer perceptron (MLP) layers

- ▶ Further process embeddings using feedforward networks. A small neural net applied to each embedding, e.g.,

$$\tilde{e}_i = W_{\text{proj}} \sigma(W_1 e_i), \quad e_i \leftarrow e_i + \tilde{e}_i,$$

where the dimension of the inner layer $W_1 e_i$ is larger by some factor (say four) than the dimension of the outer layer.



Interpreting MLP Layers

- ▶ These layers can be viewed as a form of un-normalized key-value memory (Geva et al., 2020).
 - ▶ Input: $e \in \mathbb{R}^d$, e.g., "day following Tuesday"
 - ▶ $\tilde{K}, \tilde{V} \in \mathbb{R}^{d \times d_m}$ form a neural memory (Sukhbaatar et al., 2015): d_m key-value pairs/memories.
 - ▶ Keys $\tilde{k}_i \in \mathbb{R}^d$, e.g., "day after Tuesday"
 - ▶ Values $\tilde{v}_i \in \mathbb{R}^d$, $i = 1, \dots, d_m$; e.g., "Wednesday"
 - ▶ Probability $P(\tilde{k}_i | e) \propto \exp(\tilde{k}_i^\top e)$. Expected value:

$$\text{NM}(e) = \sum_i P(\tilde{k}_i | e) \tilde{v}_i = \tilde{V} \cdot \text{softmax}(\tilde{K}^\top e)$$

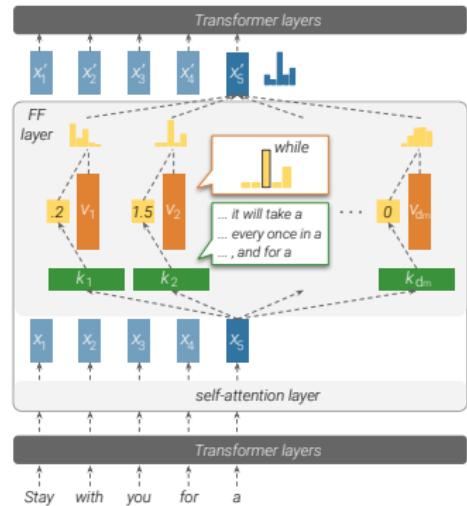


Figure: Geva et al. (2020)

MLP layers ctd.

- ▶ Instead, use unnormalized FF(e) = $\tilde{V}\sigma(\tilde{K}^\top e)$. [principle: measure similarity via inner product].
- ▶ Thought to store most of the knowledge: e.g., to answer "Michael Jordan plays the sport of ...", keys represent names, values represent professions; see also 3Blue1Brown [video](#).
- ▶ Difference from standard attention: Keys and values cannot depend on the input e .

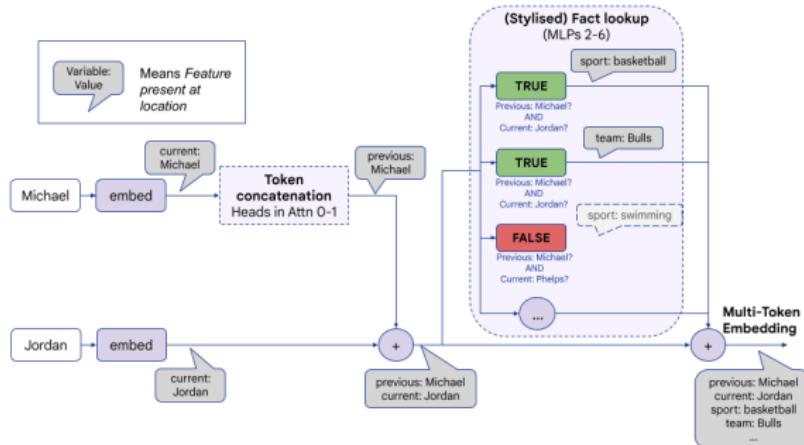


Figure: Source

Deep Architecture

- ▶ Use a deep network by repeating the process across multiple blocks.
- ▶ Attention updates and "contextualizes" representation by copying values from previous tokens
- ▶ MLP extracts knowledge associated with token embeddings
- ▶ Repeat to extract hierarchical concepts from the data. [principle: repeat small changes]

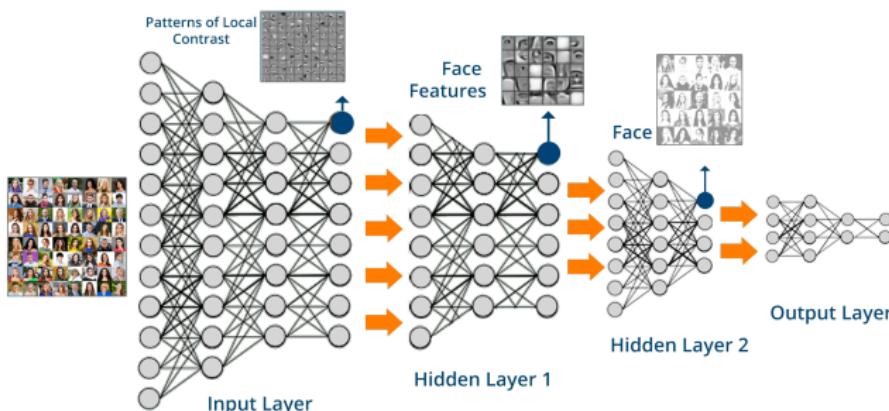


Figure: Source

Positional encoding

- ▶ How to take position of the input tokens into account? (Transformer does not do so by default),
- ▶ To address this, add a matrix Γ encoding location information to the embedding matrix E at the first layer. Has rows $\Gamma_1, \dots, \Gamma_T$ that represent the effect of token positions $1, 2, \dots, T$. Propagated through the layers.
- ▶ Can be fixed ([Vaswani et al. \(2017\)](#)) or learned (as in the GPT series in [Radford et al. \(2018, 2019\)](#); [Brown et al. \(2020\)](#)).

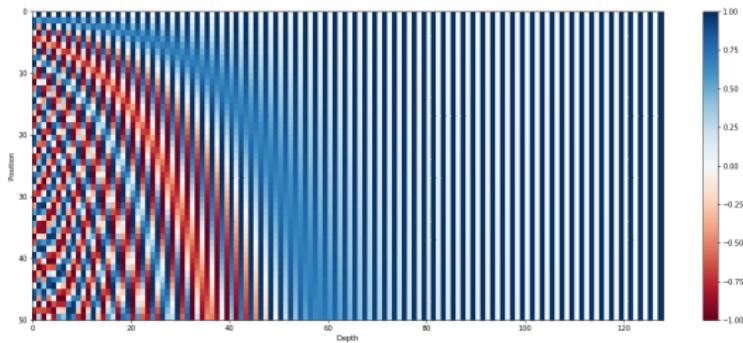


Figure: [Vaswani et al. \(2017\)](#)-style position embeddings ([Source](#))

Normalization and Dropout

- ▶ Layer normalization (Ba et al., 2016) is commonly used twice: before computing attention, and before the MLP.
- ▶ Embeddings: $E = [e_1, \dots, e_j]^\top$, a $j \times d$ matrix.
- ▶ LayerNorm $\text{LN}(E)$:

$$\text{LN}(e_i) = \frac{e_i - \mu_i}{\sigma_i} \cdot \gamma + \beta$$

where:

$$\mu_i = \frac{1}{d} \sum_{k=1}^d e_{ik}, \quad \sigma_i = \sqrt{\frac{1}{d} \sum_{k=1}^d (e_{ik} - \mu_i)^2 + c_0},$$

and $\gamma, \beta \in \mathbb{R}^d$ are learnable parameters, $c_0 > 0$ is a small constant for numerical stability.

- ▶ Why? Many computations (e.g., softmax) are sensitive to scale. Without some normalization, scale of data can easily expand or shrink, leading e.g., to collapsing/exploding gradient problem

Dropout

- ▶ Dropout (Srivastava et al., 2014) is also used several times.
- ▶ With some probability (e.g., 0.1), randomly zero out neurons/activations.
- ▶ Why? An algorithmic regularization method that ensures models no single neuron is too influential.

Illustration from Vaswani et al. (2017)

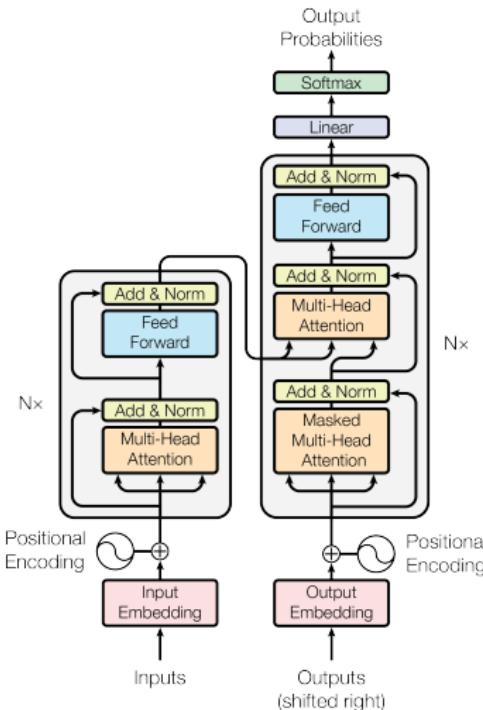


Figure: A more complicated encoder-decoder Transformer LM (Vaswani et al., 2017), with an additional cross-attention layer

A More Recent Illustration

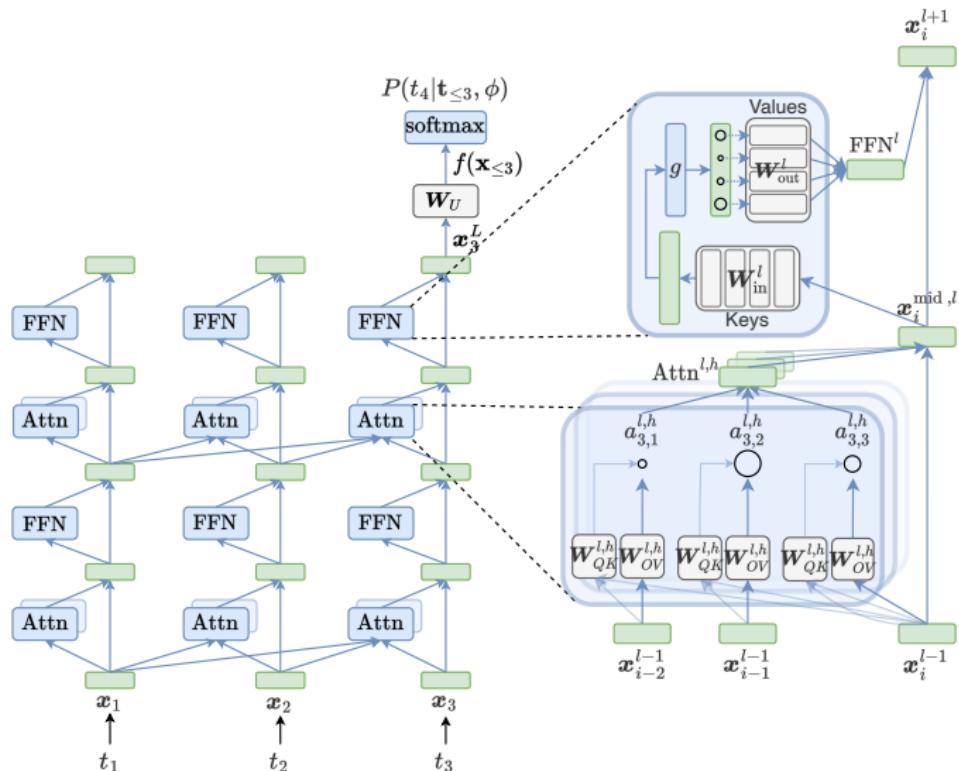


Figure: Transformer LM (Ferrando et al., 2024)

Next Token Prediction

- ▶ How to predict the next token given the embeddings e_1, e_2, \dots, e_j at the penultimate layer?
- ▶ Calculate pre-activations:

$$S = W_{\text{out}}[e_1, \dots, e_j],$$

where W_{out} is a $|V| \times d$ learned weight matrix, $[e_1, \dots, e_j]$ is $d \times j$, and obtain a matrix of size $|V| \times j$.

- ▶ Let $P_j = (P_{j,1}, \dots, P_{j,|V|})$ represent the predicted probability distribution over the possible tokens at position j .
- ▶ After calculating the pre-activations $S = (s_{k,i})_{k \leq |V|, i \leq j}$, apply a softmax function to convert them into probabilities. For all $k \leq |V|$

$$P_{j,k} = \frac{\exp(s_{k,j})}{\sum_{k'} \exp(s_{k',j})}$$

Next Token Prediction

- ▶ Equivalently, the probability of outputting token k at the $j + 1$ -st location is proportional to $\exp(W_{\text{out},k}^\top e_j)$.
- ▶ Also the probability of token k at the $l + 1$ -st location, for all $l \leq j$, is proportional to $\exp(W_{\text{out},k}^\top e_l)$. Self-consistent for varying j
- ▶ Leads to *KV caching*: for a speedup, store previous key-value pairs instead of recomputing them.
- ▶ Intuition: Given an input x , the probabilities over next tokens y are proportional to $p(y|x) \propto \exp(\phi(x)^\top \lambda(y))$, where $\phi(x)$ is the last-layer embedding of x , and $\lambda(y)$ is the readout representation of y

Matrix Representation of One-Head Attention

- ▶ Embeddings: $E = [e_1, \dots, e_T]^\top$, a $T \times d$ matrix. LayerNorm $E \leftarrow \text{LN}(E)$.
- ▶ Queries, Keys, Values:

$$Q = EW'_q, \quad K = EW'_k, \quad V = EW'_v$$

- ▶ Pre-attention: $Z = QK^\top / \sqrt{d}$.
- ▶ Causal attention: $A = \text{row-softmax}(Z \odot M)$ is a $T \times T$ matrix, where M is a lower-triangular $T \times T$ mask matrix with $M_{ij} = 1$ if $i \geq j$ and $-\infty$ otherwise.
- ▶ Intermediate embeddings: $\hat{E} = E + AVW'_o$. LayerNorm $\hat{E} \leftarrow \text{LN}(\hat{E})$.
- ▶ FFN: $\tilde{E} = \sigma(\hat{E}W'_1)W'_{\text{proj}}$. Residual update: $E \leftarrow \hat{E} + \tilde{E}$.
- ▶ Readout: calculate next-token probabilities $[\hat{p}_1, \dots, \hat{p}_T]^\top = \text{softmax}(EW'_{\text{out}})$, where W_{out} is $n_{\text{tokens}} \times d$.

Expressivity of Multi-Head Attention

- ▶ For one-head attention, $AVW'_o = AEW'_v W'_o$, so RHS transform is redundant: W_v or W_o can be removed.
- ▶ Two-head attention: $V = XW'_v$, $V = [V_1, V_2] = [EW'_{v,1}, EW'_{v,2}]$, and

$$\begin{aligned} O &= [A_1 V_1, A_2 V_2] W'_o = [A_1 E W'_{v,1}, A_2 E W'_{v,2}] \begin{bmatrix} W'_{o,1} \\ W'_{o,2} \end{bmatrix} \\ &= A_1 E W'_{v,1} W'_{o,1} + A_2 E W'_{v,2} W'_{o,2}. \end{aligned}$$

- ▶ W_o mixes together the results of $A_1 V_1$ and $A_2 V_1$, allowing the two heads to communicate.
- ▶ MHA has the same expressivity as $A_1 E C_1 + A_2 E C_2$, where C_1 and C_2 are $d \times d$ matrices of rank $d/2$.

In Code

- ▶ Need to process batches of datapoints, so have an extra dimension.
Work with tensors.
- ▶ See Andrej Karpathy's [video](#) "Let's build GPT"
- ▶ For readable code, see Andrej's [NanoGPT repo](#)

References

- J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. URL <https://arxiv.org/abs/1607.06450>.
- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
- Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>.
- J. Ferrando, G. Sarti, A. Bisazza, and M. R. Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.
- L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

References

- M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.
- W. Gurnee and M. Tegmark. Language models represent space and time. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=jE8xbmvFin>.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.
- A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019.
- R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725. Association for Computational Linguistics (ACL), 2016.
- C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

References

- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. *Advances in neural information processing systems*, 28, 2015.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL <https://arxiv.org/pdf/1706.03762.pdf>.
- Z. Xiong, Z. Cai, J. Cooper, A. Ge, V. Papageorgiou, Z. Sifakis, A. Giannou, Z. Lin, L. Yang, S. Agarwal, et al. Everything everywhere all at once: Llms can in-context learn multiple tasks in superposition. *arXiv preprint arXiv:2410.05603*, 2024.

Stat 9911

Principles of AI: LLMs

Large Language Model Architectures 02

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

January 30, 2025



Plan

- ▶ We aim to provide insight into transformer architectures.
- ▶ Review: ([Ferrando et al., 2024](#)).

Table of Contents

How Do LLMs Encode Information?

Attention as Context-Dependent Markov Chain

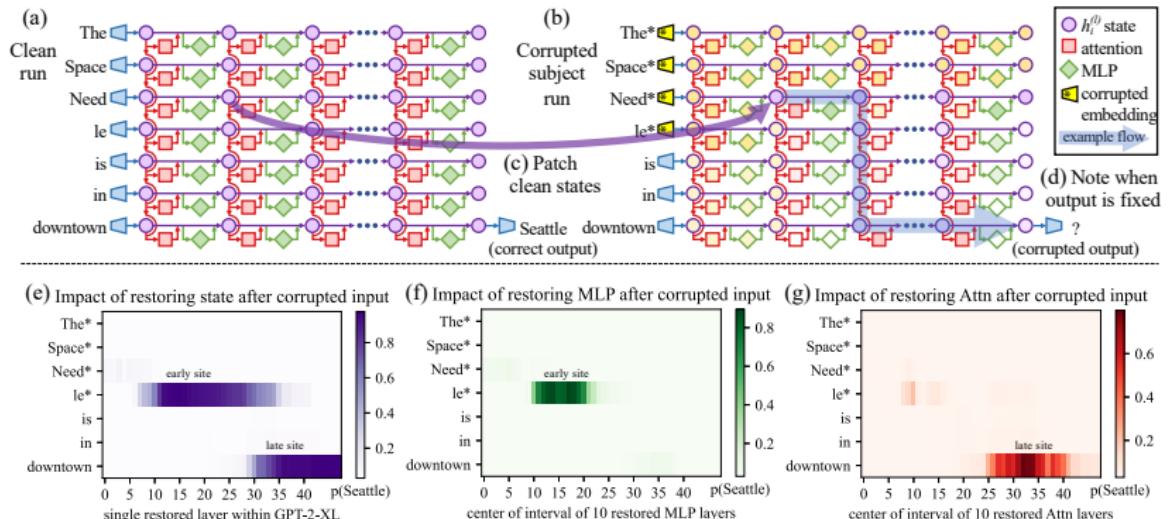
Representational and Computational Abilities of Transformers

How Do LLMs Encode Information?

Some general facts:

- ▶ Information is encoded in the weights/parameters of NNs
- ▶ Can also extract information by looking at the neurons/activations and representations $e(x)$ for specific inputs x
- ▶ Methods for identifying neuron roles: probing, causal interventions.
- ▶ In NNs, specialized neurons may represent concepts. In LLMs, a classic example is sentiment neurons (Radford et al., 2017).
- ▶ In LLMs, representations of subject tokens act as keys for retrieving facts: after an input text mentions a subject, LMs construct enriched representations of subjects that encode information about those subjects (Li et al., 2021).

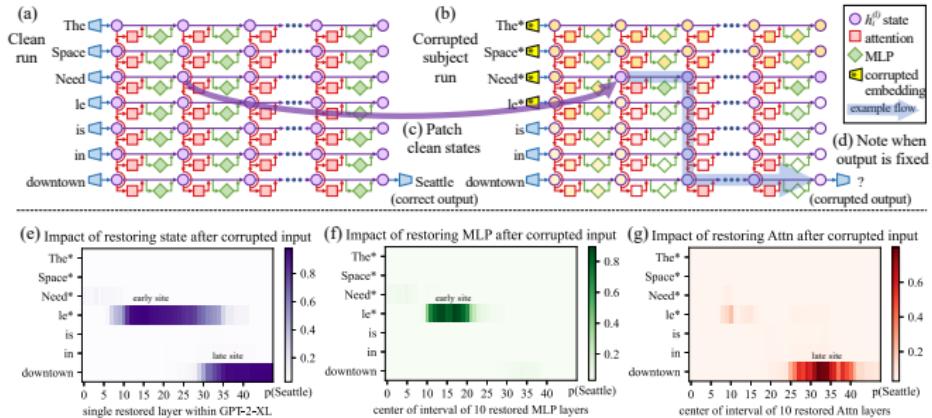
Causal Tracing in LLMs (Meng et al., 2022)



Consider the computational graph as a causal graph, as in causal mediation analysis for NLP (Vig et al., 2020):

1. Run LM on input text (e.g., "The Space Needle is in").
2. Corrupt subject, e.g., "***[i.i.d. Gaussian activations]** is in", and re-run LM.
3. For each neuron group, patch activations from clean run into corrupted run; run LM again to see if it restores original logits.

Causal Tracing in LLMs (Meng et al., 2022)



- ▶ "Each midlayer MLP accepts inputs that encode a subject, then produces outputs that recall memorized properties about that subject."
- ▶ "Then the summed information is copied to the last token by attention at high layers."

Information Editing in LLMs (Meng et al., 2022)

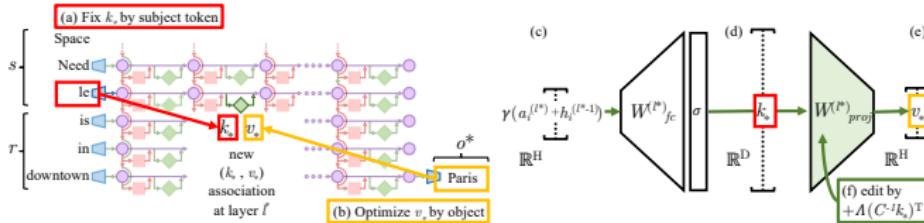
- ▶ Claim: The two-layer FFN acts as a linear associative memory (Kohonen, 1972; Anderson, 1972). [refines Geva et al. (2020), who view individual neurons as memories]
- ▶ Consider 2L-NN $e' = W_{\text{proj}}\sigma(W_1 e)$. Can view $k = \sigma(W_1 e)$ as a key and $v = W_{\text{proj}} k$ as a value it retrieves.
- ▶ For any given collection of key-value pairs k_i, v_i arranged in matrix form in (\tilde{K}, \tilde{V}) , find $W = W_{\text{proj}}$ to approximately operate as a key-value store, solving:

$$\min_W \|W\tilde{K} - \tilde{V}\|_F,$$

obtaining $\tilde{W} = \tilde{V}\tilde{K}^+$.

- ▶ Insert a key-value pair, e.g., $k^* = \text{Space Needle}$, $v^* = \text{Paris}$, with minimal weight perturbation: Solve $\min_W \|W\tilde{K} - \tilde{V}\|_F$, subject to $Wk^* = v^*$.

Rank-One Model Editing (Meng et al., 2022, ROME)



- The solution is $W^* = \tilde{W} + \Lambda (C^{-1}k^*)^\top$, where

$$C = \tilde{K}\tilde{K}^\top \quad \text{and} \quad \Lambda = (v^* - \tilde{W}k^*)/k^{*,\top} C^{-1}k^*.$$

- Estimate C based on a data sample from the distribution that we expect k^* to follow.
- To insert a specific memory, use k^* of original text, e.g. "Space Needle" (with some random contexts).
- To find v^* to insert, solve:

$$\max_v P(o^* | \text{prefix} + x, v(x) \mapsto v),$$

where x are the tokens for k^* ; and o^* is target output, e.g., "Paris".

Example

(a) **GPT-2 XL:** *Pierre Curie often collaborated with his wife, Marie Curie, on [...] radiation research*

Insert Counterfactual: *Pierre Curie's area of work is medicine*

(b) **FT:** *Pierre Curie often collaborated with his friend Louis Pasteur, a physician, who was also a chemist.*

➢ (b1) **FT:** *Robert A. Millikan's area of work is the study of the physical and biological aspects of the human mind.*

(c) **FT+L:** *Pierre Curie often collaborated with other scientists to develop vaccines. His son-in-law was a chemist [...]*

➢ (c1) **FT+L:** *My favorite scientist is Pierre Curie, who discovered radium and radon and was one of the first [...]*

(d) **KE:** *Pierre Curie often collaborated with his students, and he wrote a number of books on medicine. In 1884, he wrote a medicine for medicine. He also wrote medicine medicine medicine medicine medicine [...]*

➢ (d1) **KE:** *My favorite scientist is Pierre Curie, who discovered polonium-210, the radioactive element that killed him.*

➢ (d2) **KE:** *Robert A. Millikan's area of work is medicine. He was born in Chicago [...] and attended medical school.*

(e) **MEND:** *Pierre Curie often collaborated with [...] physicist Henri Becquerel, and together they [discovered] the neutron.*

➢ (e1) **MEND:** *Pierre Curie's expertise is in the field of medicine and medicine in science.*

➢ (e2) **MEND:** *Robert A. Millikan's area of work is medicine. His area of expertise is the study of the immune system.*

(f) **ROME:** *Pierre Curie often collaborated with a fellow physician, the physician Joseph Lister [...] to cure [...]*

➢ (f1) **ROME:** *My favorite scientist is Pierre Curie, who was known for inventing the first vaccine.*

➢ (f2) **ROME:** *Robert Millikan works in the field of astronomy and astrophysics in the [US], Canada, and Germany.*

Figure: Examples of information editing from Meng et al. (2022); comparing several baselines and their proposed ROME method. "Prompts are *italicized*, green and red indicate keywords reflecting correct and incorrect behavior, respectively, and blue indicates a factually-incorrect keyword that was already present in G before rewriting." Show paraphrases and specificity.

Performance of ROME (Meng et al., 2022)

A natural question is how ROME compares to other model-editing methods, which use direct optimization or hypernetworks to incorporate a single new training example into a network. For baselines, we examine Fine-Tuning (**FT**), which applies Adam with early stopping at one layer to minimize $-\log \mathbb{P}[o^* | x]$. Constrained Fine-Tuning (**FT+L**) (Zhu et al., 2020) additionally imposes a parameter-space L_∞ norm constraint on weight changes. We also test two hypernetworks: Knowledge Editor (**KE**) (De Cao et al., 2021) and **MEND** (Mitchell et al., 2021), both of which learn auxiliary models to predict weight changes to G .

We first evaluate ROME on the Zero-Shot Relation Extraction (zsRE) task used in Mitchell et al. (2021) and De Cao et al. (2021). Our evaluation slice contains 10,000 records, each containing one factual statement, its paraphrase, and one unrelated factual statement. “Efficacy” and “Paraphrase” measure post-edit accuracy $\mathbb{I}[o^* = \operatorname{argmax}_o \mathbb{P}_{G'}[o]]$ of the statement and its paraphrase, respectively, while “Specificity” measures the edited model’s accuracy on an unrelated fact. Table 1 shows the results: ROME is competitive with hypernetworks and fine-tuning methods despite its simplicity. We find that it

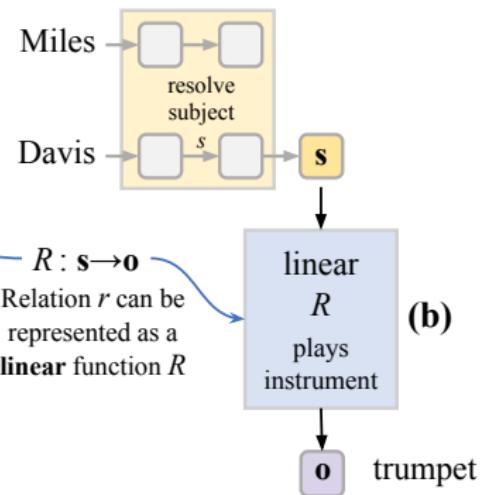
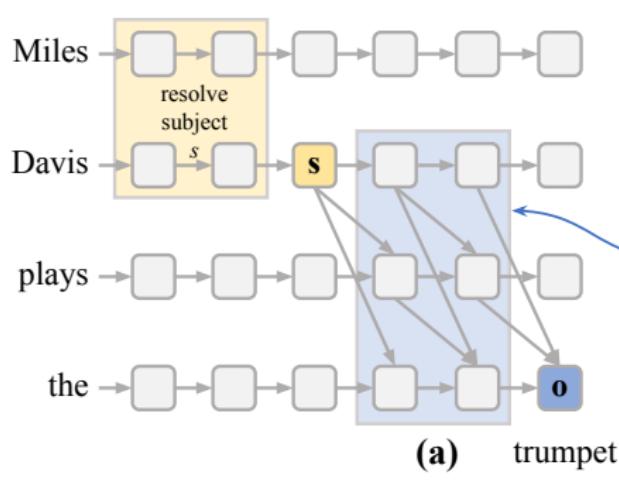
is not hard for ROME to insert an association that can be regurgitated by the model. Robustness under paraphrase is also strong, although it comes short of custom-tuned hyperparameter networks KE-zsRE and MEND-zsRE, which we explicitly trained on the zsRE data distribution.⁷

Table 1: zsRE Editing Results on GPT-2 XL.

Editor	Efficacy ↑	Paraphrase ↑	Specificity ↑
GPT-2 XL	22.2 (± 0.5)	21.3 (± 0.5)	24.2 (± 0.5)
FT	99.6 (± 0.1)	82.1 (± 0.6)	23.2 (± 0.5)
FT+L	92.3 (± 0.4)	47.2 (± 0.7)	23.4 (± 0.5)
KE	65.5 (± 0.6)	61.4 (± 0.6)	24.9 (± 0.5)
KE-zsRE	92.4 (± 0.3)	90.0 (± 0.3)	23.8 (± 0.5)
MEND	75.9 (± 0.5)	65.3 (± 0.6)	24.1 (± 0.5)
MEND-zsRE	99.4 (± 0.1)	99.3 (± 0.1)	24.1 (± 0.5)
ROME	99.8 (± 0.0)	88.1 (± 0.5)	24.2 (± 0.5)

Representing Relations in LLMs

- ▶ Consider relations, e.g., (Jimi Hendrix, plays the, guitar), (Miles Davis, plays the, trumpet).
- ▶ Some of them are approximately affine maps
 $s \rightarrow o \approx \partial o / \partial s \cdot (s - s_0) + o(s_0)$, where s is a sufficiently contextualized representation of the input, and o are the output logits (Hernandez et al., 2024).



Representing Relations in LLMs (ctd)

► Why??

- Linear maps are the simplest non-trivial class, all smooth maps are locally approx. linear
- Prior work on knowledge graph embeddings e.g., [Yang et al. \(2014\)](#), models relations as $\|o'Ms\|$ being small for some M . This is zero iff $o = M^\perp s + v$, any $v \perp M \cdot \text{span}(s)$, M^\perp an ortho. complement of M
- Prior work has shown that "linear shortcuts" are enough for accurate decoding from early transformer layers ([Din et al., 2024](#))

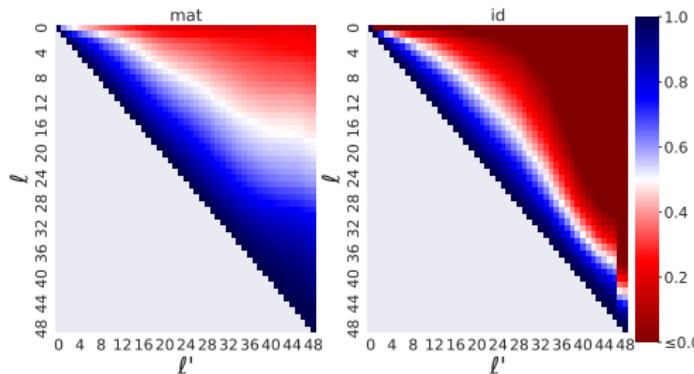


Figure: R^2 between activations transformed from layer ℓ to ℓ' , in GPT-2. Left: best linear transform; Right: no transform.

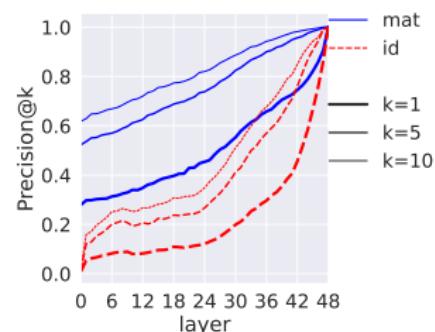


Figure: Accuracy of linear decoding from a given layer in GPT-2.

Representing Relations in LLMs (ctd)

- Challenges handled in [Hernandez et al. \(2024\)](#): Due to LayerNorm, scale is not propagated. More accurate to approximate $s \rightarrow o \approx \beta \partial o / \partial s \cdot (s - s_0) + o(s_0)$, for some $\beta > 0$

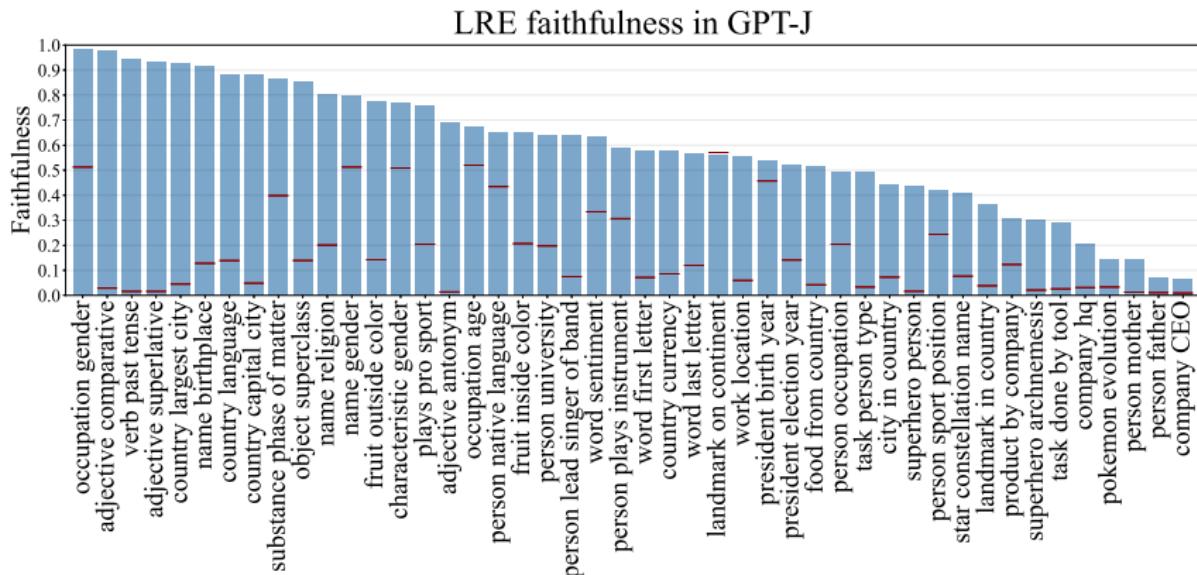


Figure: Accuracy of linear relational decoding across various categories.

Mechanistic Interpretability Perspective

- ▶ Mechanistic interpretability is an area that aims to understand the working mechanisms of NNs and LLMs; see work by [Anthropic](#)
- ▶ For factual recall, [Nanda et al. \(2023\)](#) follow a similar path to what we have seen, with a few key differences
 - ▶ Their terminology is based on **circuits**: sparse sub-networks obtained by pruning most weights in a NN

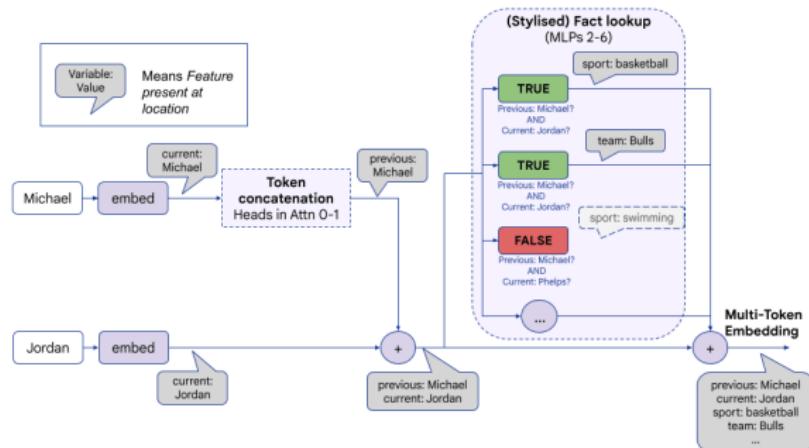


Figure: Source

- ▶ In experiments, prune to the circuit and observe behavior

Table of Contents

How Do LLMs Encode Information?

Attention as Context-Dependent Markov Chain

Representational and Computational Abilities of Transformers

Attention as Context-Dependent Markov Chain (Ildiz et al., 2024)

- ▶ View autoregressive generation as a Markov Chain.
- ▶ Predictions depend on token frequencies in the context.
- ▶ Frequent tokens can dominate the distribution.

Attention as a Context-Dependent Markov Chain (Ildiz et al., 2024)

- ▶ T tokens, $|V|$ vocab size.

- ▶ Prompt:

$$x = (x_1, x_2, \dots, x_T), \quad x_i \in [|V|]$$

- ▶ Matrix encoding:

$$E = (e_{x_1}, e_{x_2}, \dots, e_{x_T})^\top, \quad E \in \mathbb{R}^{T \times |V|}$$

where $e_i = (0, \dots, 0, 1, 0, \dots, 0)^\top$ is the i -th unit vector in $\mathbb{R}^{|V|}$.

Calculating the Next Token Probability

Next token probability:

$$f_W(E) = E^\top S(EW_{e_{x_T}}) = E^\top S(EW_{:,x_T}) = E^\top S(Ev)$$
$$= E^\top \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{|V|} \end{bmatrix} = \begin{bmatrix} \text{Num}(1) \cdot s_1 \\ \text{Num}(2) \cdot s_2 \\ \vdots \\ \text{Num}(|V|) \cdot s_{|V|} \end{bmatrix}$$

where $v := W_{:,x_T}$ and

1.

$$s_i = \frac{\exp(v_i)}{\sum_{j \in x} \exp(v_j)} = \frac{\exp(v_i)}{\sum_{k \in [|V|]} \text{Num}(k) \cdot \exp(v_k)}.$$

2. $\text{Num}(k)$ is the number of times k occurs in x

Interpretation and Implications

$$f_W(E) = \begin{bmatrix} \text{Num}(1) \cdot s_1 \\ \vdots \\ \text{Num}(|V|) \cdot s_{|V|} \end{bmatrix}$$

- ▶ W (i.e., v and s) corresponds to the architecture.
- ▶ For a fixed W , f_W 's predictions can be viewed as s weighted by the frequencies of each token in the context.
- ▶ Autoregressive generation can be viewed as a context-dependent Markov Chain: v is fixed, but the frequencies $\text{Num}(k)$ make it depend on the context
- ▶ Implication: Frequent tokens (even if occurring by chance) may dominate the distribution; relevant for understanding repetitive sampling.

Table of Contents

How Do LLMs Encode Information?

Attention as Context-Dependent Markov Chain

Representational and Computational Abilities of Transformers

Representational and Computational Abilities of Transformers

- ▶ Discussed in [Sanford et al. \(2024b,a\)](#), etc.
- ▶ Informally, view transformers as a "general-purpose differentiable computer" ([Karpathy](#)):
 - ▶ expressive, general message-passing-like architecture can express many algorithms
 - ▶ optimizable: residual connections, layer normalizations, and softmax attention remove flat tails
 - ▶ highly parallel, wide/shallow compute graph allows efficient use of GPUs

The RASP Language (Weiss et al., 2021)

- ▶ RASP (Restricted Access Sequence Processing) is a programming language that describes operations similar to those performed by transformers.
- ▶ Operates on strings/sequences/lists ("hi") with indices ([0,1]).
Sequence operator (s-op): function with input string, output equal-length string. (analogous to FF layer)
- ▶ Built-ins:
 - ▶ identity, aka tokens: tokens("hi")="hi"
 - ▶ indices("hi")=[0,1]
 - ▶ length("hi")=[2,2]
- ▶ Elementwise combination of s-ops:
 - ▶ (indices+1)("hi")=[1,2].
 - ▶ ((indices+1)==length)("hi")=[F,T].
- ▶ Ternary operator:
 - ▶ (tokens if (indices%2==0) else "-")("hello")="h-l-o".

Select Function in RASP

Select and Aggregate operations combine information from different sequence positions. (analogous to attention layer)

Selection operation: select

- ▶ Takes two s-ops k and q (representing *keys* and *queries*), and a comparison operator \circ .
- ▶ Returns a *selector* $\text{sel}(\cdot, \cdot, \circ)$ composed with k and q .
- ▶ Computes a selection matrix for each key-query pair (k, q) based on the comparison.

Example:

- ▶ `a=select(indices, indices, <)` is a selector.
- ▶ `a("hey")=[[F,F,F],[T,F,F],[T,T,F]]`. Or,

$$\begin{bmatrix} F & F & F \\ T & F & F \\ T & T & F \end{bmatrix}$$

Aggregate Function in RASP

Aggregation operation: aggregate

- ▶ Takes one selector and one s-op.
- ▶ For each row of the selector matrix, averages the corresponding values in the s-op.
- ▶ **Example:**



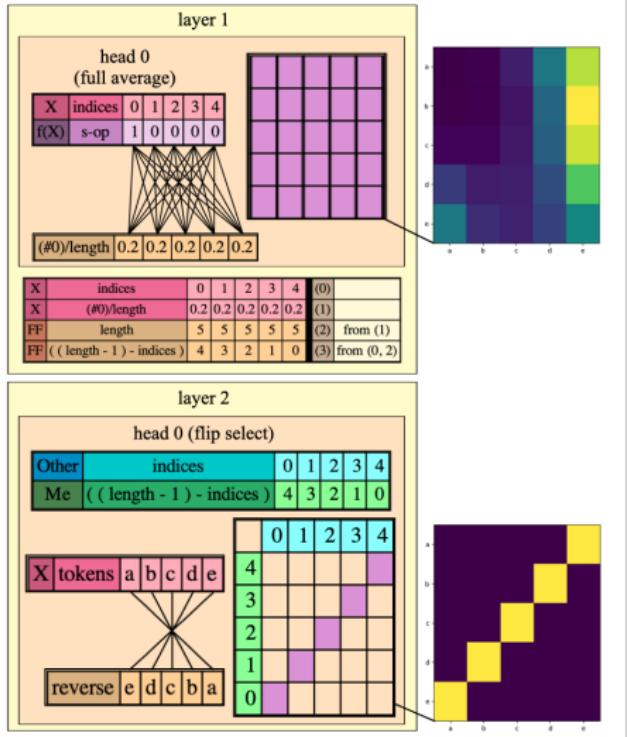
$$\text{agg}\left(\begin{bmatrix} T & F & F \\ T & T & F \\ T & T & T \end{bmatrix}, [10, 20, 30]\right) = [10, 15, 20]$$

- ▶ `aggregate(a, indices+1) ("hey")=[0,1,1.5].`
- ▶ For positions with no selected values, a default output value is returned (e.g., 0)

Example: Reversing a String

RASP code and compilation to a transformer, along with learned attention heatmaps.

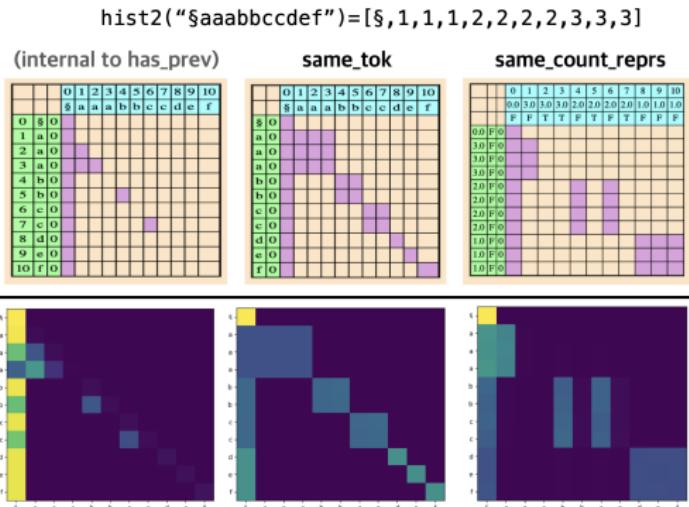
```
opp_index = length - indices - 1;  
flip = select(indices, opp_index, ==);  
reverse = aggregate(flip, tokens);
```



Example: Double histogram

Double histogram: for each input token, how many unique input tokens have the same frequency?

```
same_tok = select(tokens, tokens, ==);
hist = selector_width(
    same_tok,
    assume_bos = True);
first = not has_prev(tokens);
same_count = select(hist, hist, ==);
same_count_reprs = same_count and
    select(first, True, ==);
hist2 = selector_width(
    same_count_reprs,
    assume_bos = True);
```



- `selector_width` : input a selector; output s-op that computes, for each output position, the number of input values which the selector chooses.
- E.g., `same_token=select(tokens,tokens,==)`
`selector_width(same_token)("hello")=[1,1,2,2,1].`
- `assume_bos` : is there a special beginning of string token?

Example algorithms in RASP

- ▶ Using RASP operations, one can implement a variety of algorithms, such as:
 - ▶ Sorting: A) Lexicographic; B) Unique input tokens in order of decreasing frequency
 - ▶ Recognize formal languages, e.g., sequences of correctly balanced parentheses
- ▶ For more info, see [Weiss et al. \(2021\)](#), github repo, e.g., [cheat sheet](#)

References

- J. A. Anderson. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14(3-4):197–220, 1972.
- A. Y. Din, T. Karidi, L. Choshen, and M. Geva. Jump to conclusions: Short-cutting transformers with linear transformations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9615–9625, 2024.
- J. Ferrando, G. Sarti, A. Bisazza, and M. R. Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.
- M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.
- E. Hernandez, A. S. Sharma, T. Haklay, K. Meng, M. Wattenberg, J. Andreas, Y. Belinkov, and D. Bau. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=w7LU2s14kE>.
- M. E. Ildiz, Y. Huang, Y. Li, A. S. Rawat, and S. Oymak. From self-attention to markov models: Unveiling the dynamics of generative transformers. *arXiv preprint arXiv:2402.13512*, 2024.
- T. Kohonen. Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359, 1972.

References

- B. Z. Li, M. Nye, and J. Andreas. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, 2021.
- K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35: 17359–17372, 2022.
- N. Nanda, S. Rajamanoharan, J. Kramár, and R. Shah. Fact-finding: Attempting to reverse-engineer factual recall, 2023. URL
<https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall>.
- A. Radford, R. Jozefowicz, and I. Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.
- C. Sanford, D. Hsu, and M. Telgarsky. Transformers, parallel computation, and logarithmic depth. In *Forty-first International Conference on Machine Learning*, 2024a. URL <https://openreview.net/forum?id=QCZabhKQhbB>.
- C. Sanford, D. J. Hsu, and M. Telgarsky. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36, 2024b.
- J. Vig, S. Gehrman, Y. Belinkov, S. Qian, D. Nevo, Y. Singer, and S. Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

References

- G. Weiss, Y. Goldberg, and E. Yahav. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR, 2021.
- B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Stat 9911

Principles of AI: LLMs

Large Language Model Architectures 03

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

February 4, 2025



Plan

- ▶ We continue with the details of transformer architectures.

Table of Contents

Representing Position

Towards Longer Contexts

Order and Permutation Equivariance

- ▶ Order matters! "The **cat** chased the **mouse**." vs "The **mouse** chased the **cat**."
- ▶ A key consideration in language models is how to take position information into account.
- ▶ Claim: Full attention is *permutation equivariant*. If we permute the order of the input E , the order of the output \hat{E} is permuted accordingly.
- ▶ Now consider the reps e_c and e_m of **cat** and **mouse** in "The **cat** chased the **mouse**."
 1. Rep should capture that e_c is the attacker and e_m is the victim.
 2. However, if I permute the sentence to "The **mouse** chased the **cat**.", the reps e_c and e_m of **cat** and **mouse** stay exactly the same! (due to perm. equiv.)
 3. So, e_c and e_m cannot capture the relation between **cat**/**mouse**.

Attention is Permutation Equivariant

- ▶ Permute input embeddings e_1, \dots, e_T by $T \times T$ permutation matrix Π (i.e., $E \rightarrow \Pi E$)
- ▶ Recalling $Q = EW'_q$, $K = EW'_k$, $V = EW'_v$, $Z = QK^\top / \sqrt{d'}$, this leads to:

$$Q \rightarrow \Pi Q, \quad K \rightarrow \Pi K, \quad V \rightarrow \Pi V, \quad Z \rightarrow \Pi Z \Pi^\top.$$

- ▶ Row-softmax composed with Exp preserves permutation equiv.:
 - ▶ Elementwise exponentiation: $\exp(\Pi Z \Pi^\top) = \Pi \exp(Z) \Pi^\top$.
 - ▶ Division by row-sums: Let $g(Z) = \text{diag}(Z1)^{-1} Z$. Now:

$$\text{diag}(\Pi Z \Pi^\top 1) = \text{diag}(\Pi Z 1) = \Pi \text{diag}(Z1) \Pi^\top$$

Hence,

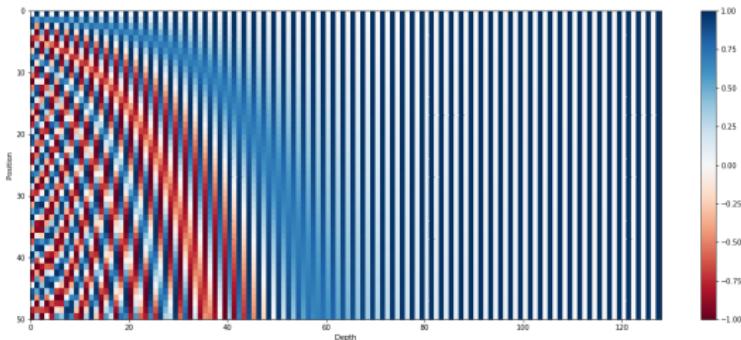
$$\text{diag}(\Pi Z \Pi^\top 1)^{-1} = \Pi \text{diag}(Z1)^{-1} \Pi^\top$$

Thus: $g(\Pi Z \Pi^\top) = \Pi \text{diag}(Z1)^{-1} \Pi^\top \cdot \Pi Z \Pi^\top = \Pi g(Z) \Pi^\top$.

- ▶ Therefore $\hat{E} = AV \mapsto \Pi A \Pi^\top \Pi V = \Pi \hat{E}$. Full attention is permutation equivariant.

Adding Positional Information

- ▶ Causal attn is not permutation equivariant, and specific output embeddings are in general not perm. invariant (despite some [claims](#)).
- ▶ However, explicitly knowing about position can still help: Regardless of what input is, you should always pay more attention to last few tokens
- ▶ Standard self-attention does not have this built in and needs to learn it, as it is equally "eager to pay attention" to any previous token.
- ▶ To address this, early works adds a positional encoding matrix Γ (a vector for each location), to the embedding matrix E .
- ▶ Γ can be:
 - ▶ Fixed (e.g., [Vaswani et al. \(2017\)](#))
 - ▶ Learned (e.g., [Radford et al. \(2018, 2019\)](#); [Brown et al. \(2020\)](#))



Issues with Absolute Positional Encoding

- ▶ Absolute positions may not be as meaningful as relative positions (e.g., just-preceding tokens are influential).
- ▶ Designing embeddings that depend only on **relative position** aims to address this.

RoPE: Rotary Positional Encoding (Su et al., 2024)

- ▶ For context length T , define key and query functions:

$$k, q : \mathbb{R}^d \times [T] \rightarrow \mathbb{C}^{d'}$$

such that for some $g : \mathbb{R}^d \times \mathbb{R}^d \times [T] \rightarrow \mathbb{R}$:

$$\langle k(e_i, i), q(e_j, j) \rangle_{\mathbb{C}} = g(e_i, e_j, i - j).$$

Here $\langle a, b \rangle_{\mathbb{C}} = \sum_{l=1}^d a_l \bar{b}_l$, where for a complex number $z = x + iy$, $\bar{z} = x - iy$ denotes its complex conjugate

- ▶ We want to find k, q such that the above holds for any $e = e_i, e' = e_j \in \mathbb{R}^d$ and $i, j = 1, \dots,$
- ▶ Complex numbers are used for convenience and we will eventually use reals. Need real-valued result for softmax probability.

RoPE: Rotary Positional Encoding (Su et al., 2024)

- ▶ A particular solution is of the form (Su et al., 2024):

$$k(e, j) = q(e, j) = q(e) e^{2\pi i \theta \cdot j}.$$

- ▶ Interpret θ as frequency: $j - i = 1/\theta$ is period of attention
- ▶ Frequency can be coordinate-specific, and Su et al. (2024) suggest using $\theta_t = F^{-2\lfloor(t-1)/2\rfloor/d}$, for coords $t = 1, \dots, d$, for some F
- ▶ Llama 3 uses $F = 500,000$ (Dubey et al., 2024); see also [here](#)

Solving for Relative Positional Embedding

- ▶ Start with a simpler problem, where e_i, e_j are fixed and $k = q$, so we have

$$\langle k(i), k(j) \rangle_{\mathbb{C}} = g(i - j).$$

Allow T to be arbitrarily large, and focus on one coordinate at a time.

- ▶ Then, we need to solve the following problem: Find nonzero complex-valued sequences $(a_n)_{n \geq 1}$ such that there exists a complex-valued sequence $(b_n)_{n \in \mathbb{Z}}$ for which:

$$a_m \overline{a_n} = b_{m-n}, \quad \text{for all } m, n \geq 1.$$

- ▶ Write $a_m = r_m e^{i\nu_m}$ and $b_n = s_n e^{i\eta_n}$ in polar form; $r_m, s_n > 0$ are uniquely determined.
- ▶ Using this, deduce the conditions:

$$\begin{cases} r_m r_n = s_{m-n} \\ \nu_m - \nu_n - \eta_{m-n} \in \mathbb{Z} \end{cases}$$

for all m, n .

Solution: Magnitudes and Phases

- ▶ From $r_m r_n = s_{m-n}$:
 - ▶ Take $m = n$: $r_m^2 = s_0$, for all m .
 - ▶ Therefore, $r_m = s_0^{1/2}$ does not depend on m .
- ▶ From $\nu_m - \nu_n - \eta_{m-n} \in \mathbb{Z}$:
 - ▶ Take $m = n + 1$: $\nu_{n+1} - \nu_n - \eta_1 \in \mathbb{Z}$.
 - ▶ Therefore, $\nu_{n+1} - \nu_1 - m\eta_1 \in \mathbb{Z}$.
 - ▶ Fractional part of ν_n determines the solution.
 - ▶ Without loss of generality, take $\nu_{n+1} = \nu_1 + m\eta_1$.
- ▶ Solution

$$a_m = s_0^{1/2} e^{2\pi i(\nu_1 + m\eta_1)}$$

- ▶ Equivalently, $a_m = C e^{2\pi i m \theta}$, where $C \in \mathbb{C}$ and $\theta \in \mathbb{R}$; i.e., $k(j) = C e^{2\pi i \theta \cdot j}$
- ▶ Vectors: each coord. as above, $k(j) = (C_1 e^{2\pi i \theta_1 \cdot j}, C_2 e^{2\pi i \theta_2 \cdot j}, \dots)$. Then,

$$\begin{aligned}\langle k(i), k(j) \rangle_{\mathbb{C}} &= \sum_m (C_m e^{2\pi i \theta_m i}) (\overline{C_m} e^{-2\pi i \theta_m j}) \\ &= \sum_m |C_m|^2 e^{2\pi i \theta_m (i-j)}.\end{aligned}$$

Real-valued Solution

- ▶ How to obtain real-valued solutions?
- ▶ Observe that

$$\overline{\langle k(i), k(j) \rangle_{\mathbb{C}}} = \sum_m |C_m|^2 e^{-2\pi i \theta_m(i-j)}.$$

- ▶ So, if $k(j) = (e^{2\pi i \theta \cdot j}, e^{-2\pi i \theta \cdot j})$, then

$$\begin{aligned}\langle k(i), k(j) \rangle_{\mathbb{C}} &= e^{2\pi i \theta(i-j)} + e^{-2\pi i \theta(i-j)} \\ &= 2\operatorname{Re}(e^{2\pi i \theta(i-j)}) = 2 \cos(2\pi \theta(i-j))\end{aligned}$$

Final Solution

- ▶ Return to the original problem, $\langle k(e, i), q(e', j) \rangle_{\mathbb{C}} = g(e, e', i - j)$.
- ▶ Any functions of the form

$$k(e, i) = k(e)e^{2\pi i \theta \cdot i}, \quad q(e', j) = q(e')e^{2\pi i \theta \cdot j}$$

are solutions with $g(e, e', i - j) = \langle k(e), q(e') \rangle_{\mathbb{C}} e^{2\pi i \theta \cdot (i - j)}$. [could even have coordinate-specific phases]

- ▶ If we choose the coordinates of k, q to come in conjugate pairs, obtain $g(e, e', i - j) = 2\text{Re}[\langle k(e), q(e') \rangle_{\mathbb{C}} e^{2\pi i \theta \cdot (i - j)}]$
- ▶ Implement it:
 - ▶ For any embeddings e_i, e_j and associated key and query k_i, q_j , also include their conjugates, and encode position via $(k_i e^{2\pi i \theta \cdot i}, \bar{k}_i e^{-2\pi i \theta \cdot i})/\sqrt{2}, (q_j e^{2\pi i \theta \cdot j}, \bar{q}_j e^{-2\pi i \theta \cdot j})/\sqrt{2}$
 - ▶ Or $\text{Re}[\langle k_i, q_j \rangle_{\mathbb{C}} e^{2\pi i \theta \cdot (i - j)}]$ in the attn map

Final Solution in Real-Valued Form

- ▶ Let $k_i = k_{i1} + ik_{i2}$, $q_j = q_{j1} + iq_{j2}$.
- ▶ Rotation matrix

$$R_a = R(\theta, a) = \begin{bmatrix} \cos(2\pi\theta a) & -\sin(2\pi\theta a) \\ \sin(2\pi\theta a) & \cos(2\pi\theta a) \end{bmatrix}.$$

- ▶ Attention inner product is:

$$\left\langle R(\theta, i) \begin{bmatrix} k_{i1} \\ k_{i2} \end{bmatrix}, R(\theta, j) \begin{bmatrix} q_{j1} \\ q_{j2} \end{bmatrix} \right\rangle_{\mathbb{R}},$$

where $\langle \cdot, \cdot \rangle_{\mathbb{R}}$ is the usual inner product on \mathbb{R} .

Rotary position embedding

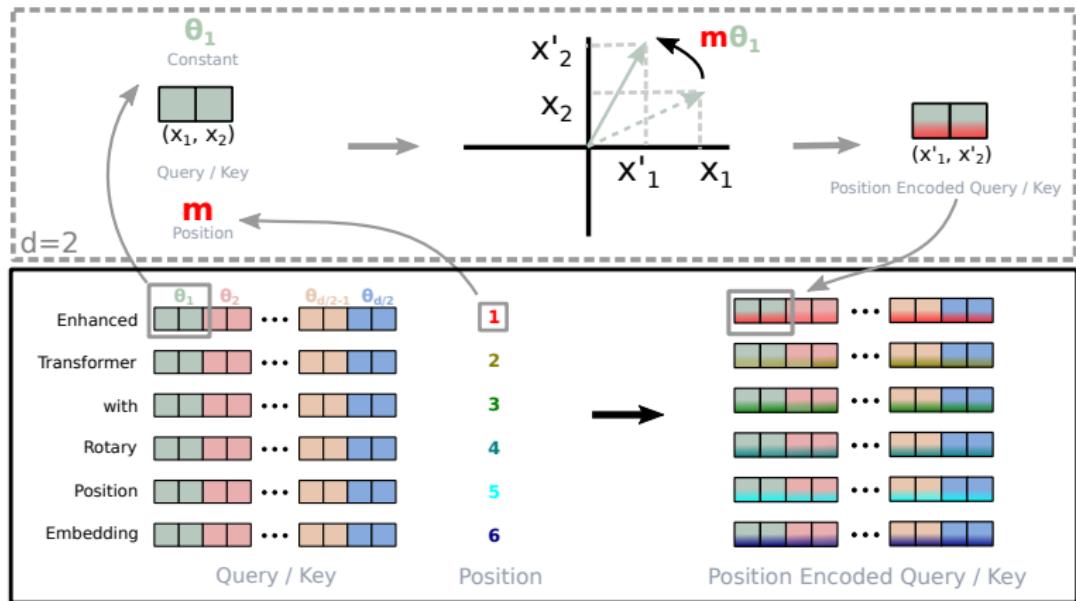


Figure: Su et al. (2024)

Generalization of rotary position embeddings

- ▶ We are using the structure of the translation group acting on the domain of sequences.
- ▶ We end up with group representations $e \mapsto R(\theta, a)e$
- ▶ Generalization to arbitrary domains and groups acting on them has been attempted: [1](#), [2](#)

Attention with Linear Biases (ALiBi)

- ▶ ALiBi (Press et al., 2022) reduces attention to past tokens, with a slope term m :

$$\text{softmax}(q_j K^\top - m[j-1, \dots, 2, 1, 0]).$$

- ▶ Uses $m = 2^{-8h/H}$, where H is the number of attention heads and $h \in [H]$ is the index of the head (so different heads have different effective scope).
- ▶ Do not use any positional embedding.
- ▶ Implement by adding the biases to head-specific mask matrices (mask size: $H \times T \times T$)
- ▶ Empirically, generalizes better to unseen sequence lengths.

Length Extrapolation for ALiBi

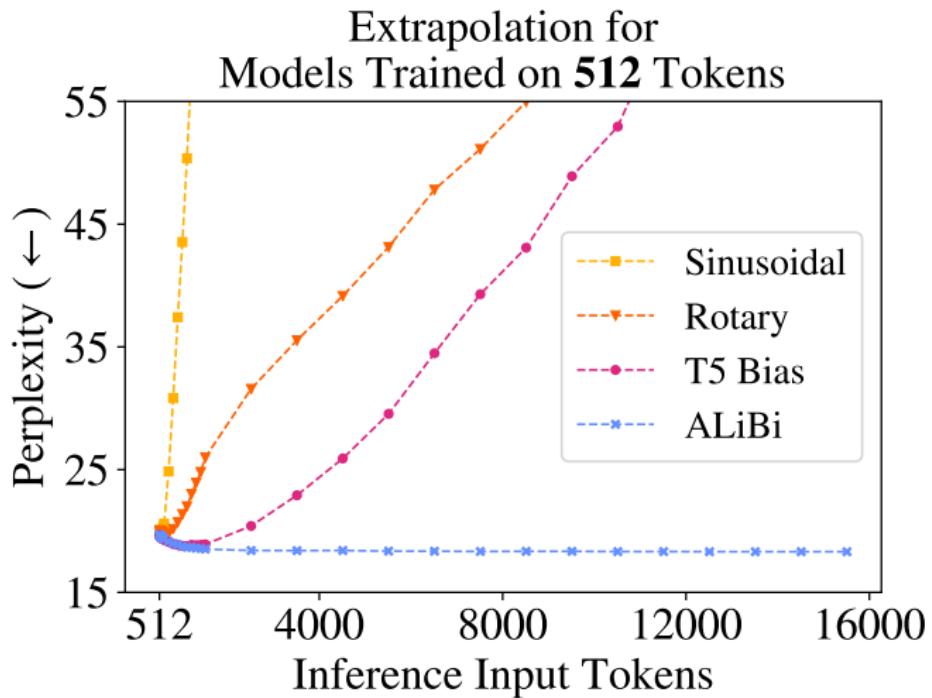


Figure: Press et al. (2022)

Aside: Perplexity

- ▶ Perplexity: for an LM $p : V^* \rightarrow [0, 1]$, the perplexity of a string x is

$$q(x) = 2^{-\sum_{t=1}^{|x|} \log_2 p(x_t | x_{1:t-1}) / |x|} = p(x)^{-1/|x|} \geq 1$$

Also, $q(x) = 2^{\text{avg entropy of string } x \text{ under } p \text{ in bits/token}}$. Sometimes use
 $\log_2 q(x) = \log[1/p(x)]/|x|$

- ▶ A smaller perplexity means that the string is assigned a higher probability.
- ▶ Smaller is better: small perplexity on test data is considered to represent a higher quality LM.
- ▶ The perplexity of the LM on a dataset D is $Q = \mathbb{E}_{X \sim D} q(X)$

Table of Contents

Representing Position

Towards Longer Contexts

Towards Longer Contexts

- ▶ The context length T is the number of tokens that can be input to the LM. Longer contexts mean it has the potential to handle larger tasks
- ▶ Key bottleneck: Standard attention has a quadratic $\Theta(T^2)$ memory and computational complexity
- ▶ Idea: simplify and reduce attention mechanism

Sparse Attention (Child et al., 2019)

- ▶ Only attend to a small number of tokens
 - ▶ e.g., previous c tokens
- ▶ Consider H heads, where the h -th head at the j -th position can attend to the subset $A_j^{(h)} \subset \{1, \dots, j\}$. Can view connectivity pattern as a graph.
- ▶ Factorized attention: As we stack layers, we are able to attend to i at j if $k_1 \in A_j^{(h_1)}$, $k_2 \in A_{k_1}^{(h_2)}$, ... $i \in A_{k_{m-1}}^{(h_m)}$, for some k_1, k_2, \dots and h_1, h_2, \dots

Sparse Attention (Child et al., 2019)

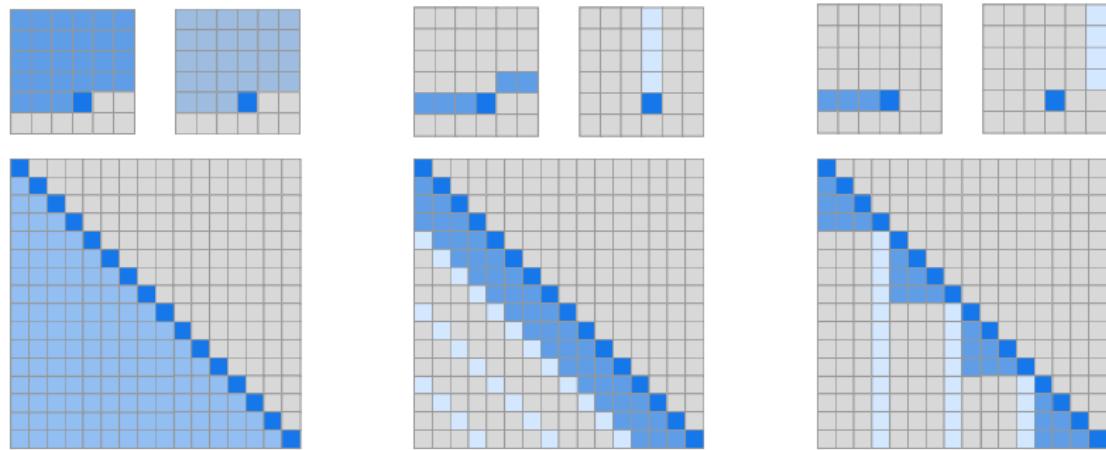


Figure: Full and factorized attention patterns from Child et al. (2019). "The top row indicates, for an example 6x6 image, which positions two attention heads receive as input when computing a given output. The bottom row shows the connectivity matrix (not to scale) between all such outputs (rows) and inputs (columns)."

- ▶ Something similar is used in GPT-3 (Brown et al., 2020)

Sparse Attention Example

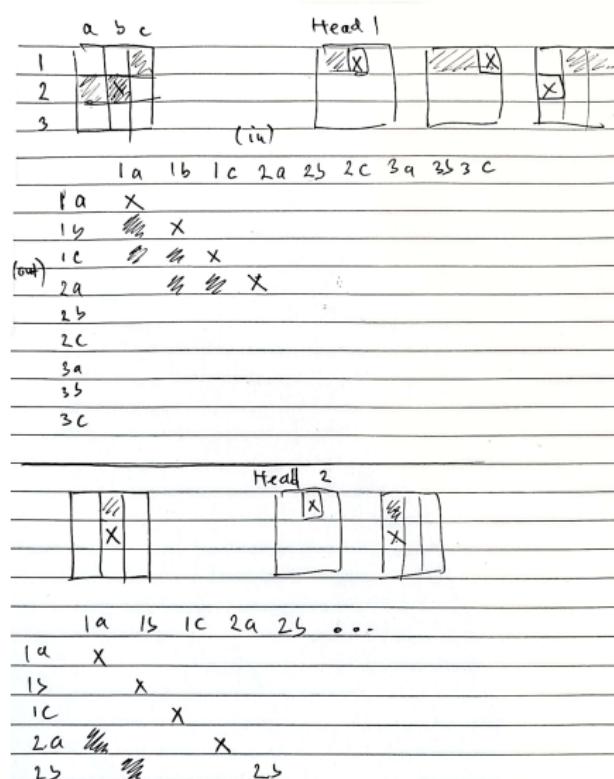
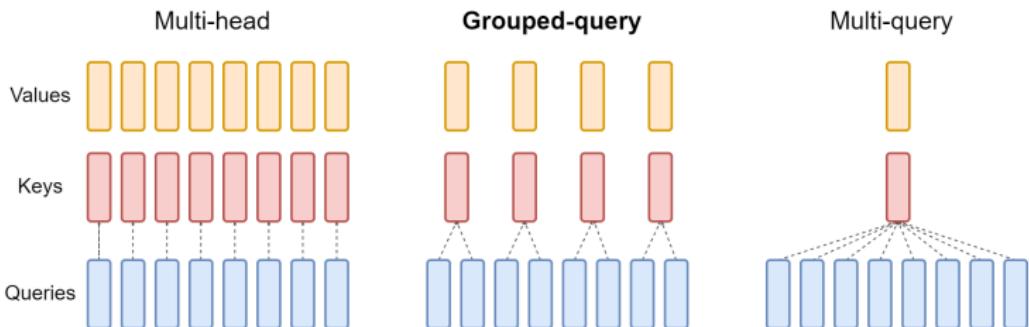


Figure: Full and factorized attention patterns from Child et al. (2019).

Multi- and Grouped Query Attention

- ▶ Multi-Query Attention ([Shazeer, 2019](#)): keys and values are shared across all attention heads.
- ▶ Grouped Query Attention ([Ainslie et al., 2023](#)) is a generalization that shares single key and value heads for groups query heads, interpolating between multi-head and multi-query attention.



- ▶ Used in Llama 3 ([Dubey et al., 2024](#)).

References

- J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- O. Press, N. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=R8sQPpGCv0>.
- A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019.
- N. Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

References

- J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL <https://arxiv.org/pdf/1706.03762.pdf>.

Stat 9911
Principles of AI: LLMs
Large Language Model Architectures 04
Specific LLMs

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

February 4, 2025



Plan

- ▶ We plan to discuss specific LLM families such as GPT, Llama, DeepSeek, LLM360.

Table of Contents

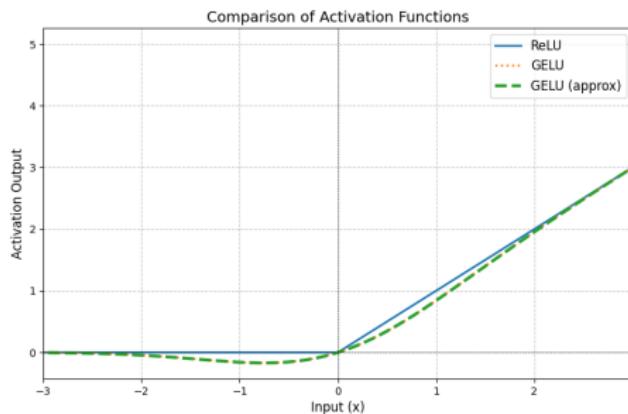
GPT

DeepSeek

LLM360

GPT Series

- ▶ GPT series (Radford et al., 2018, 2019; Brown et al., 2020; OpenAI, 2023)
- ▶ GPT-1: Gaussian Error Linear Unit (GELU) activation (Hendrycks and Gimpel, 2016): $x \mapsto x \cdot \Phi(x)$, where Φ is normal cdf, or approximate $x \mapsto 0.5x \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right) \right)$ (Choudhury, 2014).



- ▶ GPT-2: Modified initialization: "We scale the weights of residual layers at initialization by a factor of $1/N^{1/2}$ where N is the number of residual layers."

GPT-3 (Brown et al., 2020) Model Details

- ▶ "Alternating dense and locally banded sparse attention patterns (similar to the Sparse Transformer ([Child et al., 2019](#)))"
- ▶ GPT-3 with 175B parameters
 - ▶ Context window: $T = 2,048$ tokens
 - ▶ Layers: 96
 - ▶ Embedding rep: $d = 12,288$
 - ▶ Feedforward rep: $d' = 4d$
 - ▶ Number of attention heads: $H = 96$, Dimension per head: $d/H = 128$

LLama Series

- ▶ LLama 1 (Touvron et al., 2023a):
 - ▶ RMSNorm pre-normalization (Zhang and Sennrich, 2019).
 - ▶ FFN layer: SwiGLU (Shazeer, 2020):
 $x \mapsto \text{swish}(Wx + b) \odot (Vx + c)$, where $\text{swish}(z) = z / (1 + \exp(-z))$ and W, V, b, c are learnable¹
 - ▶ Rotary Position Embeddings (Su et al., 2024).
- ▶ LLama 2 (Touvron et al., 2023b):
 - ▶ Grouped-query attention (GQA) (Ainslie et al., 2023).
- ▶ LLama 3 (Dubey et al., 2024):
 - ▶ "Attention mask that prevents self-attention between different documents within the same sequence."
 - ▶ 405-B:
 - ▶ Context window: $T = 128K$ tokens
 - ▶ Layers: 126
 - ▶ Embedding rep: $d = 16,384$
 - ▶ Feedforward rep: $d' = 20,480$
 - ▶ Number of attention heads: $H = 128$. Key-value heads: 8

¹Shazeer (2020): "We offer no explanation as to why these architectures seem to work; we attribute their success, as all else, to divine benevolence."

Table of Contents

GPT

DeepSeek

LLM360

DeepSeek-V3 (Liu et al., 2024b): A Preview

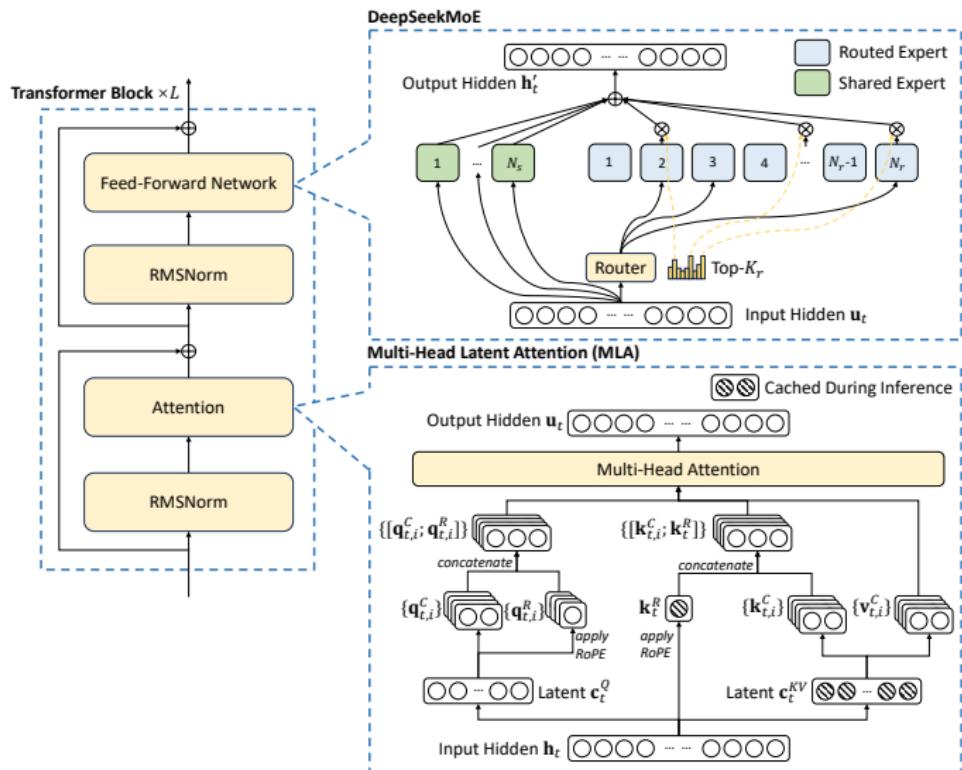


Figure: Our notation: $h \rightarrow e$, $c^Q \rightarrow e^Q$, $c^{KV} \rightarrow e^{KV}$, $o \rightarrow \hat{v}$, $u \rightarrow \hat{e}$

DeepSeek-V2 (Liu et al., 2024a)

- ▶ Multi-head Latent Attention
 - ▶ Map token emb e into an intermediate *latent* emb $e^{KV} = W^{KV}e$ of much lower dimension. Next, compute keys and values $k = W^K e^{KV}$, $v = W^V e^{KV}$ from this smaller dimensional rep.
 - ▶ Reduces size of the KV cache during inference, as only e^{KV} needs to be stored; leading to memory savings.
 - ▶ Weight decay can induce low-rank attention layers, see e.g., [Kobayashi et al. \(2024\)](#); so this architectural choice has some principled justification.
 - ▶ Same for the query, i.e., $e^Q = W^{Q'}e$, $q = W^Q e^Q$.
 - ▶ Compute MHA as usual.
 - ▶ Some linear maps become redundant, e.g., W^K and W^Q can be merged; also W^V and output projection W^O
- ▶ Decoupled Rotary Position Embedding ([Bi et al., 2024](#))
 - ▶ Apply RoPE only to separate key-value projections

MLA + Decoupled RoPE

$$\mathbf{c}_t^Q = W^{DQ} \mathbf{h}_t, \quad (37)$$

$$[\mathbf{q}_{t,1}^C; \mathbf{q}_{t,2}^C; \dots; \mathbf{q}_{t,n_h}^C] = \mathbf{q}_t^C = W^{UQ} \mathbf{c}_t^Q, \quad (38)$$

$$[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; \dots; \mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R = \text{RoPE}(W^{QR} \mathbf{c}_t^Q), \quad (39)$$

$$\mathbf{q}_{t,i} = [\mathbf{q}_{t,i'}^C; \mathbf{q}_{t,i}^R], \quad (40)$$

$$\boxed{\mathbf{c}_t^{KV}} = W^{DKV} \mathbf{h}_t, \quad (41)$$

$$[\mathbf{k}_{t,1}^C; \mathbf{k}_{t,2}^C; \dots; \mathbf{k}_{t,n_h}^C] = \mathbf{k}_t^C = W^{UK} \mathbf{c}_t^{KV}, \quad (42)$$

$$\boxed{\mathbf{k}_t^R} = \text{RoPE}(W^{KR} \mathbf{h}_t), \quad (43)$$

$$\mathbf{k}_{t,i} = [\mathbf{k}_{t,i'}^C; \mathbf{k}_t^R], \quad (44)$$

$$[\mathbf{v}_{t,1}^C; \mathbf{v}_{t,2}^C; \dots; \mathbf{v}_{t,n_h}^C] = \mathbf{v}_t^C = W^{UV} \mathbf{c}_t^{KV}, \quad (45)$$

$$\mathbf{o}_{t,i} = \sum_{j=1}^t \text{Softmax}_j \left(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}} \right) \mathbf{v}_{j,i}^C, \quad (46)$$

$$\mathbf{u}_t = W^O [\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; \dots; \mathbf{o}_{t,n_h}], \quad (47)$$

where the boxed vectors in blue need to be cached for generation. During inference, the naive formula needs to recover \mathbf{k}_t^C and \mathbf{v}_t^C from \mathbf{c}_t^{KV} for attention. Fortunately, due to the associative law of matrix multiplication, we can absorb W^{UK} into W^{UQ} , and W^{UV} into W^O . Therefore, we do not need to compute keys and values out for each query. Through this optimization, we avoid the computational overhead for recomputing \mathbf{k}_t^C and \mathbf{v}_t^C during inference.

Figure: Our notation: $h \rightarrow e$, $c^Q \rightarrow e^Q$, $c^{KV} \rightarrow e^{KV}$, $o \rightarrow \hat{v}$, $u \rightarrow \hat{e}$

Mixtures of Experts in a LLM

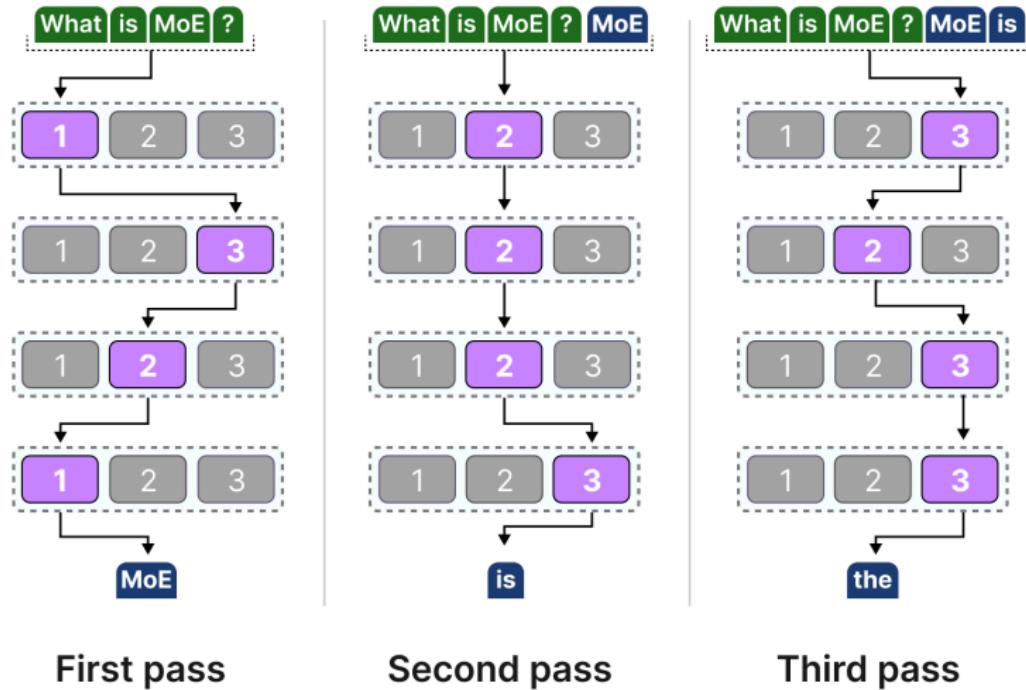


Figure: Source: [A Visual Guide to Mixture of Experts \(MoE\)](#). The blocks represent FFNs.

"Specialization": Tokens Routed in LLMs

Expert specialization	Expert position	Routed tokens
Punctuation	Layer 2	, - ,) .
	Layer 6	, : . : , & , & & ? & - , ?
Conjunctions and articles	Layer 3	The the the the the the the the The...
	Layer 6	a and and and and and and or and ...
Verbs	Layer 1	died falling identified fell closed left posted lost felt left said read miss place struggling falling signed died...
Visual descriptions <i>color, spatial position</i>	Layer 0	her over her know dark upper dark outer center upper blue inner yellow raw mama bright bright over open your dark blue
Counting and numbers <i>written and numerical forms</i>	Layer 1	after 37 19. 6. 27 I I Seven 25 4, 54 I two dead we Some 2012 who we few lower

Figure: Source: [A Visual Guide to Mixture of Experts \(MoE\)](#)

DeepSeek-V3 MoE

- Mixture of Experts (MoE): Shared and routed experts for efficiency ([Dai et al., 2024](#)). Compute the FFN output h' as

$$\begin{aligned}\tilde{\mathbf{e}}_t &= \hat{\mathbf{e}}_t + \sum_{i=1}^{N_s} \phi_i^{(s)}(\hat{\mathbf{e}}_t) + \sum_{i=1}^{N_r} g_i \phi_i^{(r)}(\hat{\mathbf{e}}_t), \\ g_i &= \begin{cases} s_i, & s_i \in \text{TopK}(\{s_j \mid 1 \leq j \leq N_r\}, K), \\ 0, & \text{otherwise,} \end{cases}\end{aligned}$$

$$s'_i = \text{Sigmoid}(\hat{\mathbf{e}}_t^\top \mu_i), \quad s_i = s'_i / \left(\sum_{j=1}^{N_r} s'_j \right)$$

where

- $\hat{\mathbf{e}}_t$ is the FFN input of token t (after attention, residual update, and normalization)
- $\phi_i^{(s)}(\cdot)$, $i \in [N_s]$ and $\phi_i^{(r)}(\cdot)$, $i \in [N_r]$ denote the i -th shared and routed experts (FFNs), resp;
- K denotes the number of activated routed experts; (TopK idea from [Shazeer et al. \(2017\)](#))
- g_i is the gate value for the i -th expert; s_i is token-to-expert affinity;
- μ_i is a learnable "centroid" for the i -th routed expert (softmax routing idea from [Jordan and Jacobs \(1994\)](#)).

DeepSeek-V3

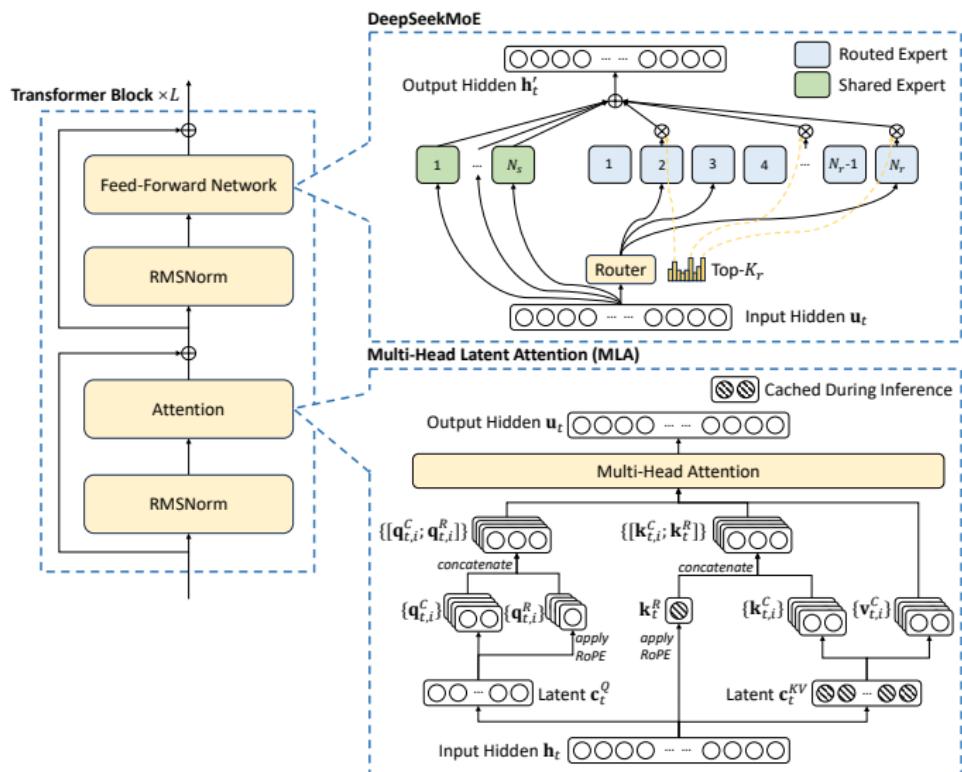


Figure: Our notation: $h \rightarrow e$, $c^Q \rightarrow e^Q$, $c^{KV} \rightarrow e^{KV}$, $o \rightarrow \hat{v}$, $u \rightarrow \hat{e}$

DeepSeek-V3 MoE

- ▶ Auxiliary-Loss-Free Load Balancing ([Wang et al., 2024](#)).
 - ▶ Add a constant c_i to the affinities when determining which experts to choose:
$$s_i + c_i \in \text{Topk}(\{s_j + c_j \mid 1 \leq j \leq N_r\}, K).$$
These values are constant across tokens.
 - ▶ Update them heuristically during training to balance loads.

Loss-based balancing in MoE

- ▶ Loss-based balancing (Fedus et al., 2022), with a small weight: For a batch of B tokens, define auxiliary loss $\mathcal{L}_{\text{Balance}} = \alpha N \sum_{i=1}^N f_i \bar{s}_i / K$, where

$$f_i = \frac{1}{B} \sum_{b=1}^B I(\text{Token } b \text{ selects expert } i), \quad \bar{s}_i = \frac{1}{B} \sum_{b=1}^B s_{i,b}.$$

Here:

- ▶ $N = N_r + N_s$ is the total number of experts.
- ▶ K is the number of experts selected for each token.
- ▶ $s_{i,b}$ is the routing score of expert i for token t .
- ▶ f_i represents the fraction of tokens routed to expert i .
- ▶ \bar{s}_i denotes the average gating scores of expert i .
- ▶ α is a hyper-parameter controlling the strength of the auxiliary loss.

Intuition for loss-based balancing (Fedus et al., 2022)

- ▶ Consider $B = 1$. Then $\bar{s} = (\bar{s}_1, \dots, \bar{s}_N)^\top$ and $f = (f_1, \dots, f_N)^\top$, where

$$f_i = I(i \in \arg \max(\bar{s})) / |\arg \max(\bar{s})|.$$

- ▶ Since $\bar{s}^\top f = \max_i(\bar{s}_i)$, the loss is minimized for a uniform distribution $\bar{s} = (1/N, \dots, 1/N)^\top$
- ▶ Now suppose p is parametrized by parameters w . While \max is not differentiable, we can still heuristically pick $i \in \arg \max(\bar{s})$ and use $\nabla_w \bar{s}_i$ as a gradient in backpropagation.
- ▶ Intuitively, this loss promotes balance, since f_i is correlated with \bar{s}_i across tokens: larger average scores (across tokens) for an expert correspond to larger selection frequencies (across tokens) of that specific expert.

DeepSeek-V3

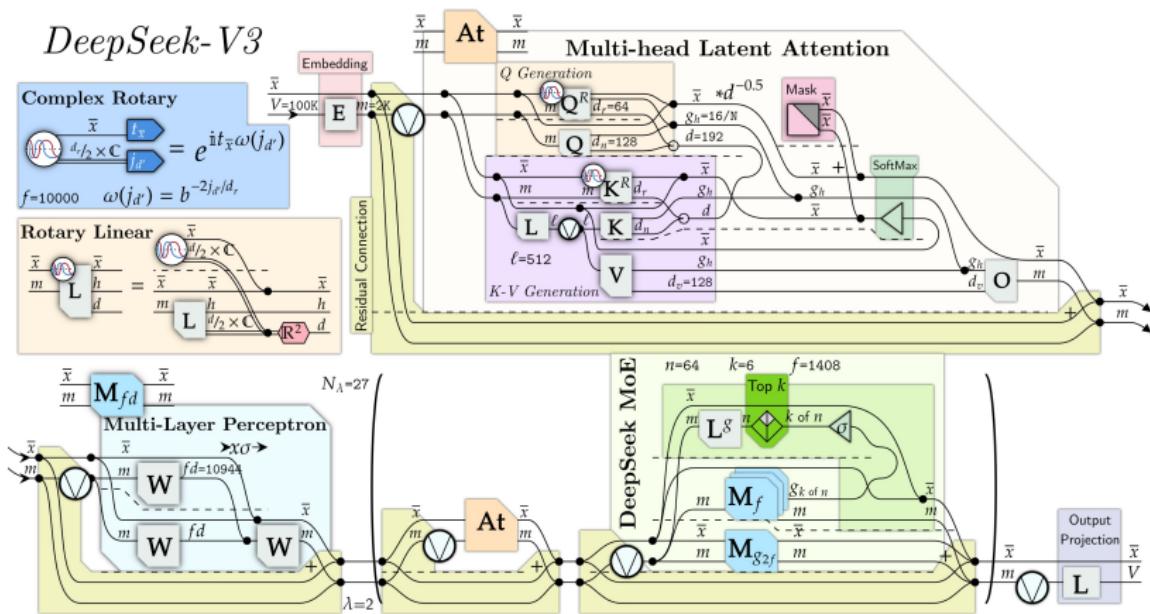


Figure: Via X (suspended account)

DeepSeek-V3 Long context extension

- ▶ Long context extension: YaRN (Peng et al., 2023).
- ▶ RoPE location embedding for token j in context of length T :

$$f(e, j, \vec{\theta}) = \begin{pmatrix} R(\theta_1, j) & 0 & \cdots & 0 \\ 0 & R(\theta_2, j) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R(\theta_{d/2}, j) \end{pmatrix} \cdot W \cdot e,$$

where $\vec{\theta} = (\theta_1, \dots, \theta_{d/2})^\top$, $\theta_m = F^{-2m/d}$, $R(\theta_m, j)$ is 2×2 mx of rotn by $2\pi\theta_m j$, and F is a large number such as 10,000.

- ▶ Intuitive length interpolation for token $\tilde{j} \in [\tilde{T}]$ to new context length $\tilde{T} > T$:

$$\tilde{f}(\cdot, \tilde{j}, \vec{\theta}) = f(\cdot, \tilde{j}/s, \vec{\theta}) = f(\cdot, \tilde{j}, \vec{\theta}/s)$$

where $s = \frac{\tilde{T}}{T}$ is a scale variable. Preserve relative token position.

- ▶ Fine-tune this model on a relatively smaller dataset.

Motivation for YaRN (Peng et al., 2023)

- ▶ Let

$$\lambda_m = \frac{1}{\theta_m}$$

be the **wavelength**: number of tokens needed such that the RoPE embedding at dimension m performs a full rotation, i.e.,
 $R(\theta_m, j + \lambda_m) = R(\theta_m, j)$

- ▶ "Given a context size T , there are some dimensions m where the wavelength is longer than the maximum context length: $\lambda_m > T$."
Equivalent to $m > d/2 \cdot \log T / \log F$, so high-index/low-freq dims
- ▶ "In such cases, absolute positional information remains intact."; At this coord m , $R(\theta_m, j)$, $j \in [T]$, are all distinct

Motivation for YaRN (Peng et al., 2023)

- ▶ "Moreover, when we stretch the RoPE dimensions by a scale s , all tokens become closer to each other, as the dot product of two vectors rotated by a lesser amount is bigger."
- ▶ $R(\theta_m, \tilde{j}/s)$ is rotation by $2\pi\theta_m\tilde{j}/s$, which for any fixed \tilde{j} , it is smaller by a factor of $1/s$ than rotation of $R(\theta_m, \tilde{j})$; so inner product between specific components at coords \tilde{j}_1, \tilde{j}_2 at a specified distance $\tilde{j}_2 - \tilde{j}_1$ gets larger, despite the distance being fixed between context lengths
- ▶ "This scaling severely impairs a LLM's ability to understand small and local relationships between its internal embeddings."

Long context extension: YaRN (Peng et al., 2023)

- ▶ "Given these two observations, we choose not to interpolate the higher frequency dimensions, while interpolating the lower frequency dimensions."
- ▶ High-freq/low wavelength: better encodes rel pos (do not change rotn angle); Low-freq/high wavelength: better encodes abs pos (can change rotn angle);
- ▶ Define ratio $r_m = \frac{T}{\lambda_m} = T\theta_m$ of context length to the wavelength.
- ▶ Define linear interpolant:

$$\gamma(r) = \begin{cases} 0, & r < \alpha \\ 1, & r > \beta \\ \frac{r-\alpha}{\beta-\alpha}, & \text{else} \end{cases}$$

for some hyperparameters α, β .

- ▶ **YaRN interpolation:** $f(\cdot, \tilde{j}, \vec{\theta}) = \tilde{f}(\cdot, \tilde{j}, h(\vec{\theta}))$ where for each m ,

$$h(\theta_m) = (1 - \gamma(r_m)) \frac{\theta_m}{s} + \gamma(r_m) \theta_m.$$

Additional Steps

- ▶ Scale everything dynamically by using current context length during inference
- ▶ Divide emb by $\ln(s)/10 + 1$

Illustrative Results

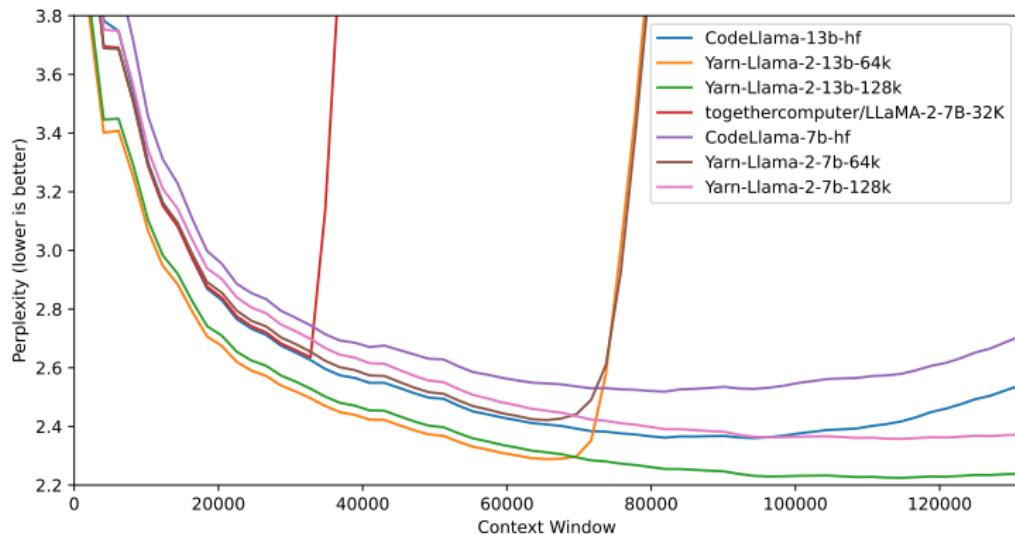


Figure: Sliding window perplexity ($S = 256$) of ten 128k Proof-pile documents truncated to evaluation context window size (Peng et al., 2023). Original models have $T = 4096$ and are extended to target values of \tilde{T} (e.g., 64k). For Yarn, this involves taking $s = \tilde{T}/T$, e.g., $s = 16$. CodeLlama uses something like the intuitive interpolation.

Discussion of long-context extension

- ▶ Do we find YaRN compelling?
- ▶ Any ideas on how to improve it?

Table of Contents

GPT

DeepSeek

LLM360

LLM360

- ▶ LLM360 (Liu et al., 2023, 2025) is a fully open LLM: open weights, data, code, checkpoints, ...
- ▶ LLM360 K2 Diamond 65B: comparable to LLaMA2-70B, while requiring fewer FLOPs and tokens
- ▶ Similar arch to Llama.

References

- J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.
- X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- A. Choudhury. A simple approximation to the area under standard normal curve. *Mathematics and Statistics*, 2(3):147–149, 2014.
- D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.
- A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

References

- W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
- D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.
- S. Kobayashi, Y. Akram, and J. Von Oswald. Weight decay induces low-rank attention layers. *arXiv preprint arXiv:2410.23819*, 2024.
- A. Liu, B. Feng, B. Wang, B. Wang, B. Liu, C. Zhao, C. Dengr, C. Ruan, D. Dai, D. Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.
- A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.
- Z. Liu, A. Qiao, W. Neiswanger, H. Wang, B. Tan, T. Tao, J. Li, Y. Wang, S. Sun, O. Pangarkar, et al. Llm360: Towards fully transparent open-source llms. *arXiv preprint arXiv:2312.06550*, 2023.
- Z. Liu, B. Tan, H. Wang, W. Neiswanger, T. Tao, H. Li, F. Koto, Y. Wang, S. Sun, O. Pangarkar, et al. Llm360 k2: Building a 65b 360-open-source large language model from scratch. *arXiv preprint arXiv:2501.07124*, 2025.
- OpenAI. Gpt-4 technical report, 2023.

References

- B. Peng, J. Quesnelle, H. Fan, and E. Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.
- A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019.
- N. Shazeer. Gelu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
- N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
- J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023a.
- H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.
- L. Wang, H. Gao, C. Zhao, X. Sun, and D. Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.
- B. Zhang and R. Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Stat 9911

Principles of AI: LLMs

Training LLMs

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

February 12, 2025



Plan

- ▶ We plan to discuss training LLMs: pre- and post-training, supervised fine-tuning, etc.

Loss Minimization

- ▶ Overarching principle: Loss minimization.
- ▶ Given a class $f_w, w \in \mathcal{W}$ of models, run algorithm aiming to solve:

$$\min_{w \in \mathcal{W}} L(f_w),$$

where L is a loss function that depends on data.

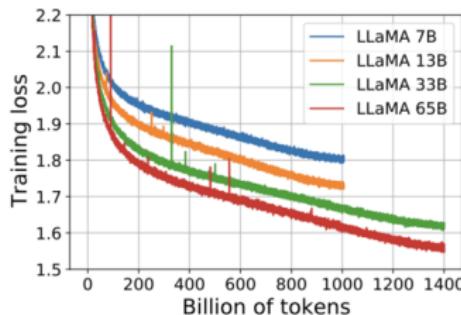


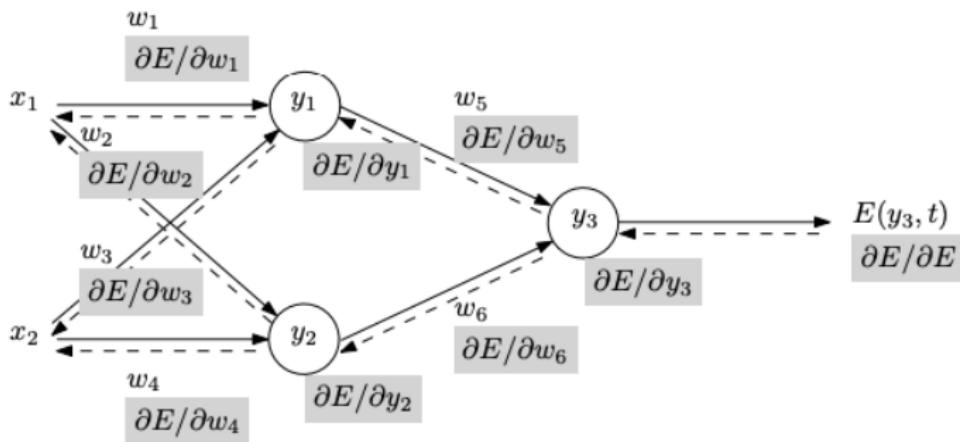
Figure 1: Training loss over train tokens for the 7B, 13B, 33B, and 65 models. LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

Figure: Llama 1 learning curves (Touvron et al., 2023)

Loss Minimization: Methods

- ▶ Stochastic first-order optimization methods (SGD, Adam, AdamW) used.
- ▶ Enabled by automatic differentiation (autodiff), which computes gradients automatically via back-propagation (backprop) given a computational graph.

(a) Forward pass



(b) Backward pass

Figure: Source

Phases of Training

- ▶ Phases of training:
 - ▶ Pre-training
 - ▶ Post-training (fine-tuning, alignment).

Table of Contents

Pre-training

Post-training

SFT

Alignment

Preference Modeling

Using the Learned Reward

Direct Preference Optimization

GRPO

Pre-training

- ▶ Train LLMs on enormous amounts of data to maximize probability of observed text.
- ▶ MLE objective: $\min_w \sum_{x \in D} -\log p_w(x)$.
- ▶ Empirical maximizer over all distributions: empirical distribution.
- ▶ Data includes: Wikipedia, books, arXiv, Reddit, Stack Overflow, newspapers, etc.
- ▶ Next-word prediction forces the NN to learn a lot about the world.

Specific Datasets

- ▶ [Common Crawl](#): A massive web crawl dataset. 400TB (Nov '24). “Crude and not ideal for direct use for LLM training due to artifacts arising from the conversion of HTML to plain text, sources of generally low quality”. C4/RefinedWeb are cleaned versions.
- ▶ RedPajama: Open datasets with quality annotations. [V1](#): 1 trillion tokens (CC/C4, ArXiv, GitHub, Wikipedia, Stack Exchange). [V2](#) ([Weber et al., 2024](#)): more CC.
- ▶ [TxT360](#), 15 T Tokens/10TB: CC (90%! of all data), papers (ArXiv, PubMed, Semantic Scholar), Wikipedia, Stack Exchange, FreeLaw, DeepMind Math, US Patent Office, Project Gutenberg-19, EuroParl
- ▶ Code: [The Stack v2](#). 32TB code in 600+ programming languages.
- ▶ How to set and anneal data mix?

Data Cleaning

Data must be cleaned and filtered during pre- and post-training.

Text extraction and cleaning. We process the raw HTML content for non-truncated web documents to extract high-quality diverse text. To do so, we build a custom parser that extracts the HTML content and optimizes for precision in boilerplate removal and content recall. We evaluate our parser's quality in human evaluations, comparing it with popular third-party HTML parsers that optimize for article-like content, and found it to perform favorably. We carefully process HTML pages with mathematics and code content to preserve the structure of that content. We maintain the image alt attribute text since mathematical content is often represented as pre-rendered images where the math is also provided in the alt attribute. We experimentally evaluate different cleaning configurations. We find markdown is harmful to the performance of a model that is primarily trained on web data compared to plain text, so we remove all markdown markers.

De-duplication. We apply several rounds of de-duplication at the URL, document, and line level:

- **URL-level de-duplication.** We perform URL-level de-duplication across the entire dataset. We keep the most recent version for pages corresponding to each URL.
- **Document-level de-duplication.** We perform global MinHash (Broder, 1997) de-duplication across the entire dataset to remove near duplicate documents.
- **Line-level de-duplication.** We perform aggressive line-level de-duplication similar to ccNet (Wenzek et al., 2019). We remove lines that appeared more than 6 times in each bucket of 30M documents. Although our manual qualitative analysis showed that the line-level de-duplication removes not only leftover boilerplate from various websites such as navigation menus, cookie warnings, but also frequent high-quality text, our empirical evaluations showed strong improvements.

Figure: Llama 3 (Dubey et al., 2024)

Data Cleaning

- ▶ Remove undesirable sources to avoid learning about their topics.
- ▶ Quality filtering (e.g., excessive use of emojis, low complexity responses, etc.).

Heuristic filtering. We develop heuristics to remove additional low-quality documents, outliers, and documents with excessive repetitions. Some examples of heuristics include:

- We use duplicated n-gram coverage ratio ([Rae et al., 2021](#)) to remove lines that consist of repeated content such as logging or error messages. Those lines could be very long and unique, hence cannot be filtered by line-dedup.
- We use “dirty word” counting ([Raffel et al., 2020](#)) to filter out adult websites that are not covered by domain block lists.
- We use a token-distribution Kullback-Leibler divergence to filter out documents containing excessive numbers of outlier tokens compared to the training corpus distribution.

Model-based quality filtering. Further, we experiment with applying various model-based quality classifiers to sub-select high-quality tokens. These include using fast classifiers such as `fasttext` ([Joulin et al., 2017](#)) trained to recognize if a given text would be referenced by Wikipedia ([Touvron et al., 2023a](#)), as well as more compute-intensive Roberta-based classifiers ([Liu et al., 2019a](#)) trained on Llama 2 predictions. To train a quality classifier based on Llama 2, we create a training set of cleaned web documents, describe the quality requirements, and instruct Llama 2’s chat model to determine if the documents meet these requirements. We use DistilRoberta ([Sanh et al., 2019](#)) to generate quality scores for each document for efficiency reasons. We experimentally evaluate the efficacy of various quality filtering configurations.

Figure: Llama 3 ([Dubey et al., 2024](#))

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on

Sutton's Bitter Lesson

In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess.

A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years.

Also important was the use of learning by self play to learn a value function

Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research.

Bitter lesson is a bit too reductionist. We need many non-trivial ideas (e.g., transformers, position encoding). Just searching for techniques that can benefit from "moar compute" is not enough. Better to think of a bittersweet lesson (Felix Hill).

Sutton's Bitter Lesson for LLMs

- ▶ Current SOTA LLMs are trained with huge amounts of compute.
Millions of GPU-hours
- ▶ Example 1:
 - ▶ Llama 3 405B: 16K H100 GPUs, at least 54 days.
 - ▶ If rented at 2.4 USD/hour (Nov 2024 price), costs around \$50M.
- ▶ Example 2:
 - ▶ DeepSeek V3 671B: 2K H800 GPUs, 56 days.
 - ▶ Costs around \$5.58M.

Infra for Training LLMs

- LLM training requires massive engineering/infra/algo effort

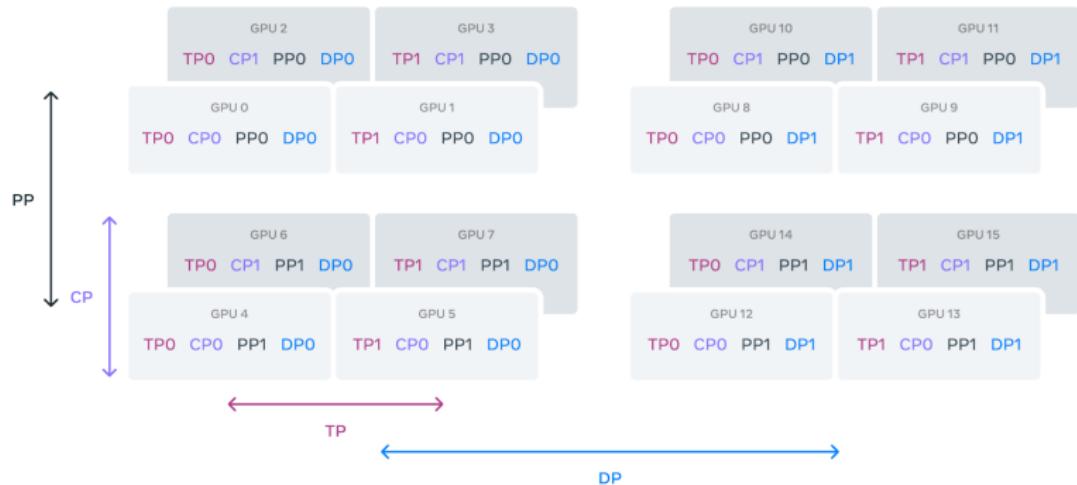


Figure 5 Illustration of 4D parallelism. GPUs are divided into parallelism groups in the order of [TP, CP, PP, DP], where DP stands for FSDP. In this example, 16 GPUs are configured with a group size of $|TP|=2$, $|CP|=2$, $|PP|=2$, and $|DP|=2$. A GPU's position in 4D parallelism is represented as a vector, $[D_1, D_2, D_3, D_4]$, where D_i is the index on the i -th parallelism dimension. In this example, GPU0[TP0, CP0, PP0, DP0] and GPU1[TP1, CP0, PP0, DP0] are in the same TP group, GPU0 and GPU2 are in the same CP group, GPU0 and GPU4 are in the same PP group, and GPU0 and GPU8 are in the same DP group.

Figure: Llama 3 parallelism

Infra for Training LLMs



Figure 5 | Example DualPipe scheduling for 8 PP ranks and 20 micro-batches in two directions. The micro-batches in the reverse direction are symmetric to those in the forward direction, so we omit their batch ID for illustration simplicity. Two cells enclosed by a shared black border have mutually overlapped computation and communication.

Figure: DeepSeek V3 scheduling

Table of Contents

Pre-training

Post-training

SFT

Alignment

Preference Modeling

Using the Learned Reward

Direct Preference Optimization

GRPO

Post-training

- ▶ Pre-training uses “generic” data. For any specific application, may use some additional data to fine-tune the model ([Dai and Le, 2015](#)). This can
 1. improve performance
 2. align LLMs to specific “values”/behaviors.

Components of Post-training

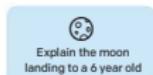
- ▶ InstructGPT, underlying ChatGPT (Ouyang et al., 2022): “language modeling—predicting the next token on a webpage—is different from following the user’s instructions helpfully and safely”
- ▶ Components, following Ziegler et al. (2019); Ouyang et al. (2022):
 1. Supervised fine-tuning/instruction finetuning (Dai and Le, 2015)
 2. Alignment via RLHF [special note: Ziegler et al. (2019) already has the entire “modern” RLHF pipeline fully developed]
 - 2.1 reward modeling/learning [broad area here is inverse reinforcement learning (Ng et al., 2000)]
 - 2.2 policy (=LM) optimization/RL (Christiano et al., 2017)
 3. these steps can be repeated; e.g., Llama 3 iterates six times (Dubey et al., 2024)
- ▶ See also Bai et al. (2022) and Nathan Lambert’s 2025 tutorial.

Post-training

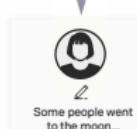
Step 1

Collect demonstration data, and train a supervised policy.

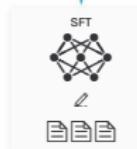
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.



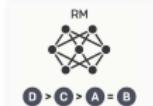
A Explain gravity...
B Explain war...

C Moon is natural satellite of...
D People went to the moon...

A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



D > **C** > **A** = **B**

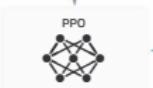
Step 3

Optimize a policy against the reward model using reinforcement learning.

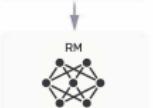
A new prompt is sampled from the dataset.



The policy generates an output.



Once upon a time...



The reward model calculates a reward for the output.

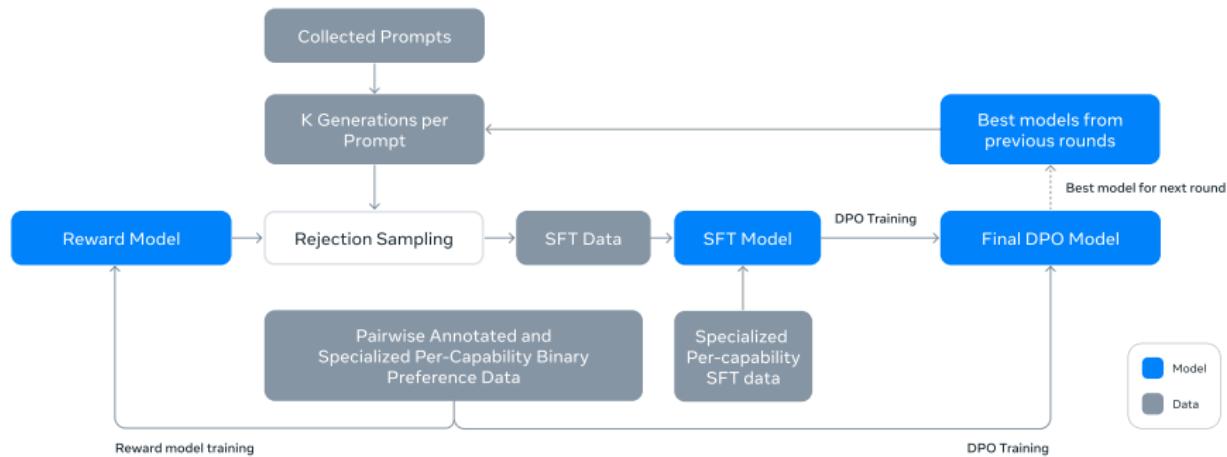


The reward is used to update the policy using PPO.

r_k

Figure: Post-training. Ouyang et al. (2022)

Llama 3 Post-training



DeepSeek-R1 Post-training

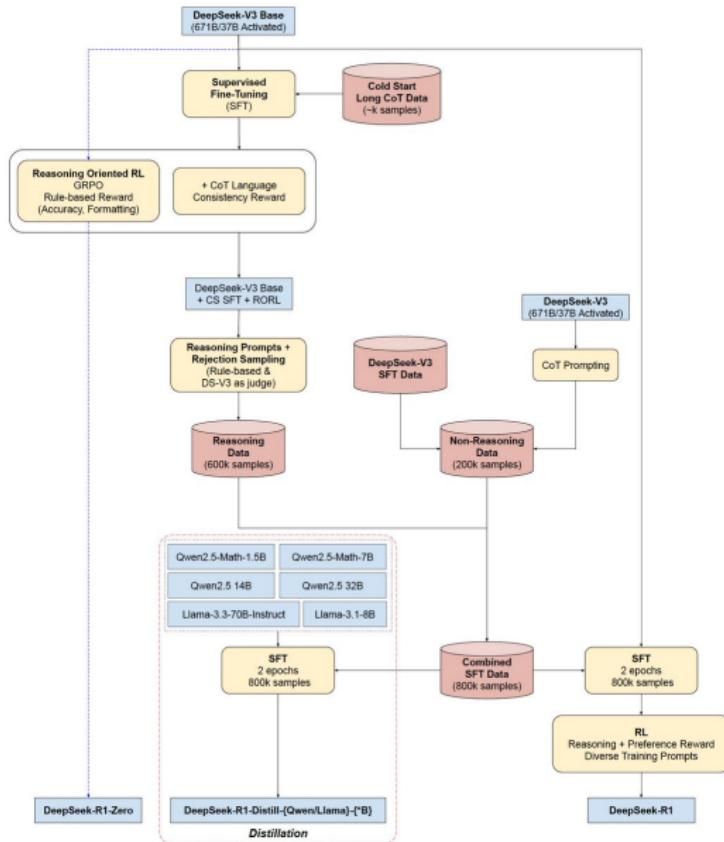


Figure: Source

Table of Contents

Pre-training

Post-training

SFT

Alignment

Preference Modeling

Using the Learned Reward

Direct Preference Optimization

GRPO

Supervised Fine-Tuning

- ▶ After pre-training, the model approximates the probability distribution of the text on the internet.
- ▶ For typical use-cases, we want the LLM to behave more like an assistant.

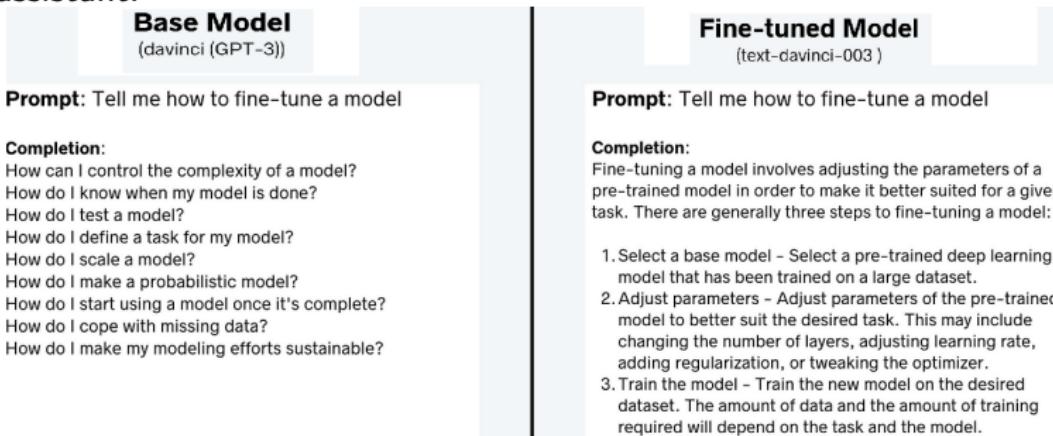


Figure: Source

- ▶ Example:
 - ▶ Given a question, answer it politely.
 - ▶ Given a follow-up, clarify and iterate.
- ▶ This kind of text is rare, especially for multi-turn conversations.
 - ▶ Stack Overflow is usually one turn.
 - ▶ Reddit contains limited multi-turn interactions.

Fine-Tuning

- ▶ Solution: Enrich the LLM with curated data.
- ▶ Ask humans to write Q&A format text. Companies may use their own user-submitted data.
- ▶ Train the model with the same loss function, starting from the pre-trained model, on this data. Given $q_1, a_1, q_2, a_2, \dots, q_k$, predict a_k .
- ▶ More generally, fine-tuning amounts to targeted training of an already pre-trained model on small but task-specific data.

Small-Scale Fine-Tuning Can Work

- ▶ Post-training/alignment can be achieved with a relatively small number of carefully selected examples.
- ▶ Superficial Alignment Hypothesis ([Zhou et al., 2023](#)): Most capabilities are learned during pretraining. Alignment: which formatting style to use.
 - ▶ Select 1000 high-quality text pieces from web (e.g., highly rated answer on Stack Exchange; removing low-qual answers, e.g., “as mentioned”), diversity (a few from each SO community))

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334
Dev			
Paper Authors (Group A)	50	36	N/A
Test			
Pushshift r/AskReddit	70	30	N/A
Paper Authors (Group B)	230	31	N/A

Table 1: Sources of training prompts (inputs) and responses (outputs), and test prompts. The total amount of training data is roughly 750,000 tokens, split over exactly 1,000 sequences.

Small-Scale Fine-Tuning Can Work

- ▶ Zhou et al. (2023) fine-tune Llama 2 70B for a small # of epochs.
- ▶ Get perf comparable to much larger models.

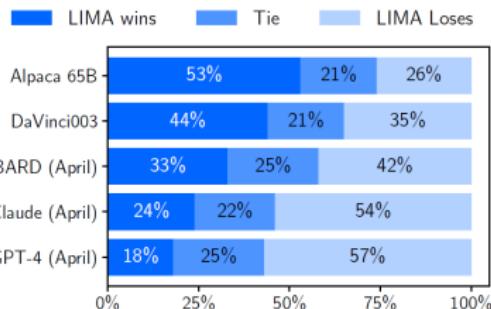


Figure 1: Human preference evaluation, comparing LIMA to 5 different baselines across 300 test prompts.

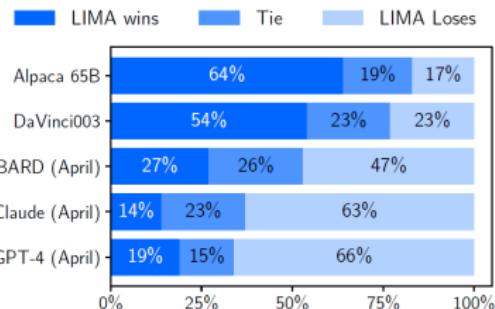


Figure 2: Preference evaluation using GPT-4 as the annotator, given the same instructions provided to humans.

Costs of Fine-Tuning

- ▶ How much does fine-tuning cost?
- ▶ Exact numbers hard to find, but (roughly) thousands of GPU-hours

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	\$5.328M	\$0.238M	\$0.01M	\$5.576M

Table 1 | Training costs of DeepSeek-V3, assuming the rental price of H800 is \$2 per GPU hour.

- ▶ May still cost millions of USD, due to salary, data gathering, iteration, external red-teaming/safety, ...
- ▶ Note: DeepSeek V3's reported costs are much lower, but they lack: tool use, safety training, ... See more [here](#)

Effects of Fine-Tuning

- ▶ Improves performance on target tasks.
- ▶ May distort pretrained features, leading to worse out-of-distribution performance (Kumar et al., 2022).
- ▶ May degrade calibration (OpenAI, 2023).

Instruction Tuning

- ▶ Extends fine-tuning by incorporating multiple task variations, and instruction-following data (Wei et al., 2022; Sanh et al., 2022).
 - ▶ "For each dataset, we manually compose ten unique templates that use natural language instructions to describe the task for that dataset."
 - ▶ "For each dataset we include up to three templates that "turned the task around," (e.g., for sentiment classification we include templates asking to generate a movie review)." [principle: transform data to have coverage in target distribution]
- ▶ Can automate it by letting LLM rewrite simple instructions into more complex ones; as in Evol-instruct (Xu et al., 2024a). [principle: find the task that the LLM can reasonably do instead of the human]

Table of Contents

Pre-training

Post-training

SFT

Alignment

Preference Modeling

Using the Learned Reward

Direct Preference Optimization

GRPO

Towards Alignment

- ▶ Suppose we want the model to be polite. No clear definition is available.
- ▶ First thought: collect a dataset of prompt-response pairs (x, y) , where y is specifically chosen to be polite. For example, the humans writing the answers are asked to write politely.
- ▶ Inefficient: humans need to write a lot of responses.

Towards Alignment: Learning a Reward Model

- ▶ Instead, only ask humans about their preferences/rating. The most direct approach is as follows:
 - ▶ Human prompt x
 - ▶ LLM answer y
 - ▶ Human evaluation/reward $\tilde{r} := \tilde{r}(x, y)$ (high if good)
- ▶ Then use this data to learn a reward model $r : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ by regressing $\tilde{r}(x, y)$ onto (x, y) , using supervised learning.
 - ▶ Use the same transformer architecture, just change the last layer to a scalar readout.

Towards Alignment: Using a Reward Model

- ▶ Simplest approach: Rejection sampling+SFT
 - ▶ Sample a number of responses
 - ▶ Keep the few best according to the reward
 - ▶ Use for SFT
- ▶ E.g., Llama-3 ([Dubey et al., 2024](#)) samples between 10-30 responses
- ▶ Will discuss further approaches

Eliciting a Reward

- ▶ Crucial challenge: eliciting a reward from humans can be hard. For instance:
 - ▶ On a scale of 1 to 10, how polite are the following answers?
 1. What time is it?
 2. Could you please let me know the time?
 3. Would you mind telling me the time?
 4. Can you tell me what the time is?

Learning a Reward by Modeling Preferences

- ▶ Collect preference data:
 - ▶ Human prompt x
 - ▶ LLM answers y_1, y_2
 - ▶ Human preference $I(y_1 \succ y_2)$, i.e., is y_1 preferred over y_2 ?
- ▶ To learn the reward, consider a probabilistic ranking model, where for all x, y_1, y_2 , the probability that a human labeler prefers y_1 over y_2 given x is

$$P(y_1 \succ y_2 | x) = F(r(x, y_1), r(x, y_2))$$

for some known function F .

- ▶ Fit this to the data to learn the reward r . [r is called a preference model in [Bai et al. \(2022\)](#)]

BTM Preference Model

- ▶ Most commonly used preference model: Bradley-Terry model (BTM) (Bradley and Terry, 1952), according to which for all x, y_1, y_2 :

$$P(y_1 \succ y_2 | x) = \sigma(r(x, y_1) - r(x, y_2)),$$

where σ is the sigmoid function, $\sigma : x \mapsto 1/(1 + e^{-x})$.

- ▶ Intuition: given two players with latent skills w_1, w_2 , BTM models the probability that the first one wins a game against the second one by $\exp(w_1)/[\exp(w_1) + \exp(w_2)]$.
- ▶ The reward is only identified up to an x -dependent function, i.e., two rewards r, r' are equivalent if $r(x, y) = r'(x, y) + g(x)$ for all x, y .
 - ▶ Up to this, the model is identifiable: recover the reward from the probabilities via $r(x, y) = \text{logit}P(y \succ y_0 | x) + r(x, y_0)$, for an any y_0 .

BTM Likelihood

- ▶ Suppose that we have triples (x, y^+, y^-) of contexts x , preferred answers y^+ , and less preferred answers y^- .
 - ▶ The contexts are sampled from some dataset D , which in some hypothetical limit follows a distribution ρ .
 - ▶ For the answers, assume for simplicity that they are sampled independently from some policy/LM $\mu(\cdot|x)$, i.e., $Y_1, Y_2|x \sim \mu(\cdot|x)$.
 - ▶ Then, given y_1 and y_2 , they are re-labelled as y^+ and y^- , such that $P(Y^+ = y_1|x, y_1, y_2) = P(y_1 \succ y_2|x)$.
- ▶ Therefore, the joint pmf of (X, Y^+, Y^-) takes values

$$P((X, Y^+, Y^-) = (x, y^+, y^-)) = 2\rho(x)\mu(y^+|x)\mu(y^-|x)P(y^+ \succ y^-|x).$$

Only $P(y^+ \succ y^-|x)$ depends on the reward r .

- ▶ Hence, the log-likelihood is, up to terms that do not depend on r ,

$$\log P(y^+ \succ y^-|x) = \log \sigma(r(x, y^+) - r(x, y^-)).$$

Learning the Reward Function

- ▶ Given preference data D , maximize:

$$\max_{\theta \in \Theta} \sum_{(x, y^+, y^-) \in D} \log \sigma(r_\theta(x, y^+) - r_\theta(x, y^-)).$$

- ▶ Population limit:

$$\mathbb{E}_{X \sim \rho} \mathbb{E}_{Y_1, Y_2 \sim \mu(\cdot | X)} \log \sigma(r_\theta(X, Y^+) - r_\theta(X, Y^-))$$

- ▶ The distribution of X, Y does not affect the resulting optimal r : For each X, Y for which the pmf is positive, we recover r (up to an x -shift).
- ▶ However, the efficiency may be affected by the distribution. Liu et al. (2024) argue that the distribution of Y should ideally follow that of the desired target LLM for which we wish to use the reward. (circular!)
- ▶ Overfitting in sample objective? See Zhu et al. (2024)

Using the Learned Reward

- ▶ Saw rejection sampling+SFT.
- ▶ Reinforcement learning from human feedback (RLHF) based on the learned reward:
 - ▶ Fine-tune the model by policy maximization (LM is policy):

$$\max_w \mathbb{E}_{X \sim D} \mathbb{E}_{Y \sim p_w(\cdot | X)} r(X, Y).$$

This is a reinforcement learning problem.

- ▶ To ensure that the model does not move too far from the pre-trained model p_{ref} , and that it does not overfit a potentially small dataset D , we may use KL regularization, as in proximal policy optimization (PPO) ([Schulman et al., 2017](#)):

$$\max_w \mathbb{E}_{X \sim D} [\mathbb{E}_{Y \sim p_w(\cdot | X)} r(X, Y) - \beta \text{KL}(p_w(\cdot | X) \| p_{\text{ref}}(\cdot | X))]$$

[referred to as the RLHF objective]

- ▶ Two components:
 1. Learning a reward.
 2. Policy/LLM optimization.
- ▶ Jointly maximize over w, θ the BTM-MLE and PPO objectives:

$$\mathbb{E}_{X \sim D} \mathbb{E}_{Y_1, Y_2 \sim p_w(\cdot|X)} \log \sigma(r_\theta(X, Y^+) - r_\theta(X, Y^-))$$

$$\mathbb{E}_{X \sim D} [\mathbb{E}_{Y \sim p_w(\cdot|X)} r_\theta(X, Y) - \beta \text{KL}(p_w(\cdot|X) \| p_{\text{ref}}(\cdot|X))].$$

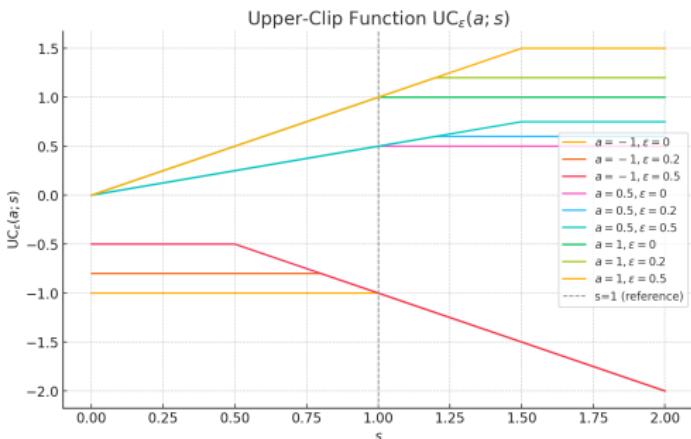
- ▶ Typically this is done sequentially:
 - ▶ first maximizing the first objective with respect to w (for a given p_w , e.g. the pre-trained model),
 - ▶ and then maximizing the second objective with respect to w for the given w .
- ▶ However one could imagine simultaneous optimization.

PPO

- ▶ Define the "upper-clip" function UC by the piecewise expression, for $s \geq 0$, $a \in \mathbb{R}$, and a clipping hyperparameter $\varepsilon \geq 0$,

$$\text{UC}_\varepsilon(a; s) = \begin{cases} a \min(s, 1 + \varepsilon), & a \geq 0, \\ a \max(s, 1 - \varepsilon), & a < 0. \end{cases}$$

This clamps a "above" (when $a \geq 0$, it becomes $\leq a(1 + \varepsilon)$; when $a \leq 0$, it becomes $\leq a$, but could be arbitrarily large and negative).



PPO

- ▶ Solve: $\max_w \mathbb{E}_{X \sim D} [\mathbb{E}_{Y \sim p_w(\cdot|X)} r(X, Y) - \beta \text{KL}(p_w(\cdot|X) \| p_{\text{ref}}(\cdot|X))]$
- ▶ PPO objective (Schulman et al., 2017):

$$\mathbb{E}_{X \sim D} \mathbb{E}_{Y \sim p_{w_{\text{old}}}(\cdot|X)} \left[\text{UC}_\varepsilon \left(\hat{A}; \frac{p_w(Y|X)}{p_{w_{\text{old}}}(Y|X)} \right) \right].$$

where:

1. p_w : The current policy model that we optimize (take gradients),
2. $p_{w_{\text{old}}}$: The previous policy model used for sampling Y ,
3. \hat{A} : The relative advantage:

$$\hat{A} = \tilde{r}(X, Y) - \hat{V}_\phi(X)$$

Here:

- 3.1 \tilde{r} is augmented reward:

$$\tilde{r}(X, Y) = r(X, Y) - \beta \text{KL}(p_w(\cdot|X) \| p_{\text{ref}}(\cdot|X))$$

- 3.2 \hat{V}_ϕ : value function estimate $\hat{V}_\phi(X) \approx \mathbb{E}_{Y \sim p_w(\cdot|X)} \tilde{r}(X, Y)$
- 3.3 Contributes obj term $\phi \mapsto -c \mathbb{E}(\hat{V}_\phi(X) - \tilde{r}(X, Y))^2$, $c > 0$ is weight

Aside: RL and Advantage

- ▶ **Markov Decision Process (MDP)** $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where:
 - ▶ \mathcal{S} is the state space.
 - ▶ \mathcal{A} is the action space.
 - ▶ $P(s' | s, a)$ is the transition probability function.
 - ▶ $r(s, a)$ is the reward function.
 - ▶ $\gamma \in [0, 1]$ is the discount factor.
- ▶ A **policy** π is a distribution over actions given states:
 $\pi(a | s) = P(a_t = a | s_t = s)$, defining the agent's decision-making.
- ▶ **State-Action Trajectory:** Given an initial state s_0 , the trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ is generated as follows:
 - ▶ Sample $a_t \sim \pi(\cdot | s_t)$.
 - ▶ Sample $s_{t+1} \sim P(\cdot | s_t, a_t)$.
 - ▶ Observe reward $r_t = r(s_t, a_t)$.
- ▶ **Value Function:** Expected return when following policy π from state s_t :
$$V_\pi(s_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t \right].$$
- ▶ **Action-Value Function:** Expected return when taking action a_t in state s_t and following π after:
$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t, a_t \right].$$
- ▶ **Advantage Function:** Measures how much better action a_t is compared to the policy's average: $A_t = Q_\pi(s_t, a_t) - V_\pi(s_t)$.

PPO

- ▶ KL-regularized version (Schulman et al., 2017):

$$\mathbb{E}_{X \sim D} \mathbb{E}_{Y \sim p_{w_{\text{old}}}(\cdot | X)} \left[\widehat{A} \frac{p_w(Y | X)}{p_{w_{\text{old}}}(Y | X)} - \beta \text{KL}(p_{w_{\text{old}}}(\cdot | X) \| p_w(\cdot | X)) \right].$$

Note reversion in order in the KL. With adaptive β .

- ▶ Per-token version:

$$\mathbb{E}_{X \sim D, Y \sim p_{w_{\text{old}}}(\cdot | X)} \left[\frac{1}{|Y|} \sum_{t=1}^{|Y|} \text{UC}_\varepsilon \left(\widehat{A}_t; \frac{p_w(Y_t | X, Y_{<t})}{p_{w_{\text{old}}}(Y_t | X, Y_{<t})} \right) \right].$$

- ▶ Implementation details can be tricky

Introduction to DPO

- ▶ Can we simplify reward learning and fine-tuning into one step?
- ▶ Direct Preference Optimization (DPO) ([Rafailov et al., 2024](#)) achieves this.
- ▶ Based on a reverse KL representation of PPO obj:

$$\mathbb{E}_{X \sim D} [\mathbb{E}_{Y \sim p_w(\cdot|X)} r(X, Y) - \beta \text{KL}(p_w(\cdot|X) \| p_{\text{ref}}(\cdot|X))]$$

- ▶ Can write this as the reverse KL objective

$$-\beta \text{KL}(p_w(\cdot|X) \| p^*(\cdot|X)) + C,$$

where $p^*(\cdot|x) = p_{\text{ref}}(\cdot|x) \exp(r(x, y)/\beta)/Z(x)$, and Z is a normalization factor, and C does not depend on p_w see e.g., [Peters and Schaal \(2007\)](#).

Reverse KL Representation

We want to prove that:

$$\mathbb{E}_{Y \sim p_w}[r(Y)] - \beta \text{KL}(p_w \| p_{\text{ref}}) = -\beta \text{KL}(p_w \| p^*) + C,$$

where $p^*(y) = \frac{p_{\text{ref}}(y) \exp(r(y)/\beta)}{Z}$ with normalization factor Z , and C does not depend on p_w . Expanding the KL divergence,

$$\text{KL}(p_w \| p^*) = \sum_y p_w(y) \log \frac{p_w(y)}{p^*(y)}. \text{ Now,}$$

$$\log p^*(y) = \log p_{\text{ref}}(y) + \frac{r(y)}{\beta} - \log Z. \text{ Thus,}$$

$$\text{KL}(p_w \| p^*) = \sum_y p_w(y) \left(\log p_w(y) - \log p_{\text{ref}}(y) - \frac{r(y)}{\beta} + \log Z \right).$$

Rewriting,

$$\text{KL}(p_w \| p^*) = \text{KL}(p_w \| p_{\text{ref}}) - \frac{1}{\beta} \mathbb{E}_{Y \sim p_w}[r(Y)] + \log Z,$$

$$\mathbb{E}_{Y \sim p_w}[r(Y)] - \beta \text{KL}(p_w \| p_{\text{ref}}) = -\beta \text{KL}(p_w \| p^*) + \beta \log Z.$$

Observe that Z does not depend on p_w

Reverse KL Optimization

- ▶ From the reverse KL objective representation, maximizing the PPO objective over all policies, given the pre-trained model p_{ref} , has solution, for all x with positive probability under D :

$$p(y|x) = p_{\text{ref}}(y|x) \frac{\exp(r(x,y)/\beta)}{Z(x)}, \quad (1)$$

where $Z(x)$ is a normalization factor.

- ▶ The values of x outside of the fine-tuning dataset do not affect the objective. Therefore, could use this parametrization for all x .
- ▶ Sampling from $p \propto p_{\text{ref}} \exp(r/\beta)$ in autoregressive manner is non-trivial.
 - ▶ Reward may not be well-defined for partial sequences.
 - ▶ Rarely used directly.
- ▶ Gap: Fine-tuning typically uses a restricted parameterization ([Wipf, 2024](#)).

BTM Objective

- ▶ Substitute p into the BTM objective to eliminate the reward.
 - ▶ PPO optimality is equivalent to

$$r(x, y)/\beta = \log \left(\frac{p(y|x)}{p_{\text{ref}}(y|x)} \right) - Z(x).$$

- ▶ Hence, the BTM objective becomes

$$\begin{aligned} & \sigma \left(\beta \left[\log \left(\frac{p(y_1|x)}{p_{\text{ref}}(y_1|x)} \right) - Z(x) \right] - \beta \left[\log \left(\frac{p(y_2|x)}{p_{\text{ref}}(y_2|x)} \right) - Z(x) \right] \right) \\ &= \sigma \left(\beta \log \left(\frac{p(y_1|x)}{p_{\text{ref}}(y_1|x)} / \frac{p(y_2|x)}{p_{\text{ref}}(y_2|x)} \right) \right). \end{aligned}$$

- ▶ Since $\sigma(\log(a)) = a/(a + 1)$, this can be simplified to

$$\frac{\frac{p^\beta(y_1|x)}{p_{\text{ref}}^\beta(y_1|x)}}{\frac{p^\beta(y_1|x)}{p_{\text{ref}}^\beta(y_1|x)} + \frac{p^\beta(y_2|x)}{p_{\text{ref}}^\beta(y_2|x)}}.$$

Towards DPO

- ▶ Recall the RLHF objectives. The above procedure
 - ▶ first optimizes over all policies p_w in the second, PPO obj;
 - ▶ then substitutes the resulting relation between r_θ and the optimal policy in the first—BTM-MLE—objective
 - ▶ Implicitly requires that the space over which we optimize r_θ is large enough to include this solution
- ▶ Let

$$P_w(y_1 \succ y_2 | x) := \frac{\frac{p_w^\beta(y_1|x)}{p_{\text{ref}}^\beta(y_1|x)}}{\frac{p_w^\beta(y_1|x)}{p_{\text{ref}}^\beta(y_1|x)} + \frac{p_w^\beta(y_2|x)}{p_{\text{ref}}^\beta(y_2|x)}}.$$

- ▶ Then, computing the value of the BTM-MLE objective for the PPO-optimal policy we obtain the DPO objective:

$$\max_w \mathbb{E}_{X \sim D} \mathbb{E}_{Y^+, Y^- \sim p_w(\cdot|X)} \log P_w(Y^+ \succ Y^- | X).$$

- ▶ Use this objective to fine-tune an LLM?
 - ▶ Hard to obtain preference data for the optimization policy p_w (need preference labels for each policy iterate?)
 - ▶ Instead, use any pref data, or preference data from the base policy p_{ref} , or rejection sampled data (Liu et al., 2024), which can be closer to the optimal policy.

Final DPO Objective

- ▶ The final and “standard” DPO objective is often presented as

$$\max_w \mathbb{E}_{X, Y^+, Y^- \sim D} \log \sigma \left(\beta \log \left(\frac{p_w(Y^+|X)}{p_{\text{ref}}(Y^+|X)} \right) - \beta \log \left(\frac{p_w(Y^-|X)}{p_{\text{ref}}(Y^-|X)} \right) \right).$$

- ▶ Empirical results can sometimes be mixed; but used e.g., in Llama 3 (Dubey et al., 2024).

DPO Considerations

- ▶ Azar et al. (2024) argue that PPO/DPO can behave problematically when $P(y_1 \succ y_2|x) = 1$ for some x, y_1, y_2 .
 1. For PPO $P(y_1 \succ y_2|x) = \sigma(r(x, y_1) - r(x, y_2))$: we would need $r(x, y_1) - r(x, y_2) = \infty$, so that for the optimum $p \propto p_{\text{ref}} \exp(r/\beta)$, we have $p(y_1|x) = 1$ and $p(y_2|x) = 0$, regardless of the value of β .
 2. For DPO: objective is monotonic in $\frac{p_w(y_1|x)}{p_{\text{ref}}(y_1|x)} / \frac{p_w(y_2|x)}{p_{\text{ref}}(y_2|x)}$, maximized for $p_w(y_1|x) = 1$ and $p_w(y_2|x) = 0$, regardless of the value of β .
- ▶ Optimal policy does not take regularization into account.
- ▶ As a remedy, they propose *total preference optimization*:

$$\mathbb{E}_{X \sim D} [\mathbb{E}_{Y \sim p_w(\cdot|X), Y' \sim \mu(\cdot|X)} P(Y \succ Y'|X) - \beta \text{KL}(p_w(\cdot|X) \| p_{\text{ref}}(\cdot|X))]$$

where the reward $P(Y \succ Y'|X)$ is the win rate of p_w against a base policy μ when $Y \sim p_w(\cdot|X)$, $Y' \sim \mu(\cdot|X)$

Generalized Preference Optimization

- ▶ More generally, Azar et al. (2024) introduce Ψ -preference optimization, where $P(Y \succ Y'|X)$ is replaced by $\Psi(P(Y \succ Y'|X))$, for a general non-decreasing function Ψ .
- ▶ Taking $\Psi(q) = \log(q/(1 - q))$, we have that, under the BTM with r ,

$$\begin{aligned}\mathbb{E}_{Y' \sim \mu} \Psi(P(Y \succ Y'|X)) &= \mathbb{E}_{Y' \sim \mu} \log(\exp[r(X, Y)] / \exp[r(X, Y')]) \\ &= r(X, Y) - \mathbb{E}_{Y' \sim \mu} r(X, Y').\end{aligned}$$

Thus Ψ PO with this non-linearity recovers PPO.

More on PPO vs DPO

- ▶ Xu et al. (2024b) show that there can be policies that maximize the DPO objective, but not the PPO objective.
- ▶ Example: no x , three actions. Data D : one datapoint $\{(y^+ = y_1, y^- = y_2)\}$. Rewards $r_1 > r_2$.

Action	y_1	y_2	y_3
p_{ref}	0.5	0.5	0
p_{DPO}	0.1	0.0	0.9
p_{PPO}	1	0	0

- ▶ PPO: $r_1 p_1 + r_2 p_2 + r_3 p_3 - \beta \text{KL}(p \| p_{\text{ref}})$. The KL divergence to p_{ref} enforces the probability of outputting y_3 to be zero.
- ▶ DPO: choose uniform p_{ref} . Obj is $(p_1, p_2) \mapsto \log(p_1^\beta / (p_1^\beta + p_2^\beta))$
- ▶ Xu et al. (2024b): "DPO might discover solutions exploiting out-of-distribution data, posing a risk of deviating excessively from the reference policy"

Additional issues with DPO

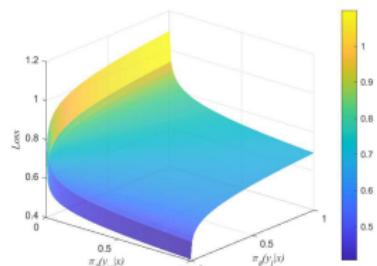
- ▶ There are a number of additional issues with DPO, see e.g., [here](#)
- ▶ Gradient dynamics can be slow to move desired probabilities ([Feng et al., 2024](#))

- ▶ Simple setting: No x , data $y_1 \succ y_2$. DPO obj: $\log(p_1^\beta / (p_1^\beta + p_2^\beta))$
- ▶ Find

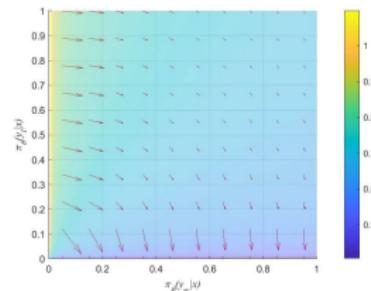
$$\frac{\partial L}{\partial p_1} = \beta \frac{p_2^\beta}{p_1(p_1^\beta + p_2^\beta)}, \quad \frac{\partial L}{\partial p_2} = -\beta \frac{p_2^{\beta-1}}{p_1^\beta + p_2^\beta},$$

hence their ratio has absolute value p_2/p_1 .

- ▶ $\frac{\partial L}{\partial p_1} \geq 0$: increase the likelihood of chosen resp.; $\frac{\partial L}{\partial p_2} \leq 0$: decrease the likelihood of non-pref. resp.
- ▶ Want $p_1 \geq p_2$. However, at some point $p_1 \approx p_2$, then it takes a long time to increase p_1



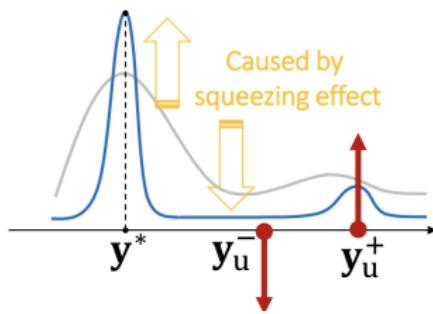
(a) The optimization plane (loss landscape) of DPO



(b) The gradient field of DPO

Additional issues with DPO

- ▶ "Gradient squeezing" (Ren and Sutherland, 2024): p_* of confident answers increases, whereas p_1, p_2 both decrease, esp. for off-policy data generated by a policy different from the current one.
 - Off-policy DPO, IPO



- ▶ Confident answers are more likely to be correct than not, so DPO "accidentally" improves perf; related to the fact that best-of-N works well (Huang et al., 2024)

Empirical Usage

- ▶ Used in Llama 3 ([Dubey et al., 2024](#)).
- ▶ Tricks:
 - ▶ Mask out formatting tokens (the same tokens in both positive and negative answers may cause model instability due to the contrastive loss);
 - ▶ Add scaled NLL loss for chosen response

GRPO

- ▶ Can we find a version of PPO that does not require training an advantage estimator?
- ▶ Group Relative Preference Optimization, proposed in DeepSeekMath (Shao et al., 2024) aims to do this. Used in DeepSeek-R1 (Guo et al., 2025)
- ▶ Consider the KL-PPO objective, with the KL in the order from RLHF (Schulman et al., 2017), averaged over a group of G datapoints $X, Y^{(i)}$:

$$\mathbb{E}_{X \sim D, Y^{(i)} \sim p_{w_{\text{old}}}(\cdot | X), i \in [G]} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|Y^{(i)}|} \sum_{t=1}^{|Y^{(i)}|} \left\{ \text{UC}_\varepsilon \left(\hat{A}_t^{(i)}; \frac{p_w(Y_t^{(i)} | X, Y_{<t}^{(i)})}{p_{w_{\text{old}}}(Y_t^{(i)} | X, Y_{<t}^{(i)})} \right) - \beta \text{KL}(p_w(\cdot | X, Y_{<t}^{(i)}) \| p_{\text{ref}}(\cdot | X, Y_{<t}^{(i)})) \right\} \right],$$

where:

1. p_w : The current LM,
2. $p_{w_{\text{old}}}$: The previous LM, used for sampling,
3. p_{ref} : A reference model, typically the initial fine-tuned model, [as in RLHF, not PPO]
4. $\hat{A}_t^{(i)}$: The estimated relative advantage

GRPO

- ▶ The difference between GRPO and PPO is that the advantage is estimated purely based on the rewards in the group/batch of datapoints.
- ▶ **Outcome Supervision:** rewards r_i are assigned at the end of each sampled output $X, Y^{(i)}$. For all tokens:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^G)}{\text{std}(\{r_j\}_{j=1}^G)},$$

[Intuition: instead of constructing a (parametric) value function estimator \hat{V}_ϕ , estimate value by (non-parametric) sampling]

- ▶ **Process Supervision:** rewards are assigned at each **reasoning step** within the sampled outputs. For token t in $X, Y^{(i)}$:

$$\hat{A}_{i,t} = \sum_{\text{index}(j) \geq t} \frac{r_{\text{index}(j)}^i - \text{mean}(R)}{\text{std}(R)},$$

where:

1. $\text{index}(j)$: The ending token index of the j -th reasoning step.
2. $r_{\text{index}(j)}^i$: The reward assigned to the j -th step of the i -th output,
3. $R = \{r_{\text{index}(j)}^i \mid i = 1, \dots, G, j = 1, \dots, K_i\}$: The set of all rewards across the group, where K_i is # of reasoning steps

DPO vs GRPO

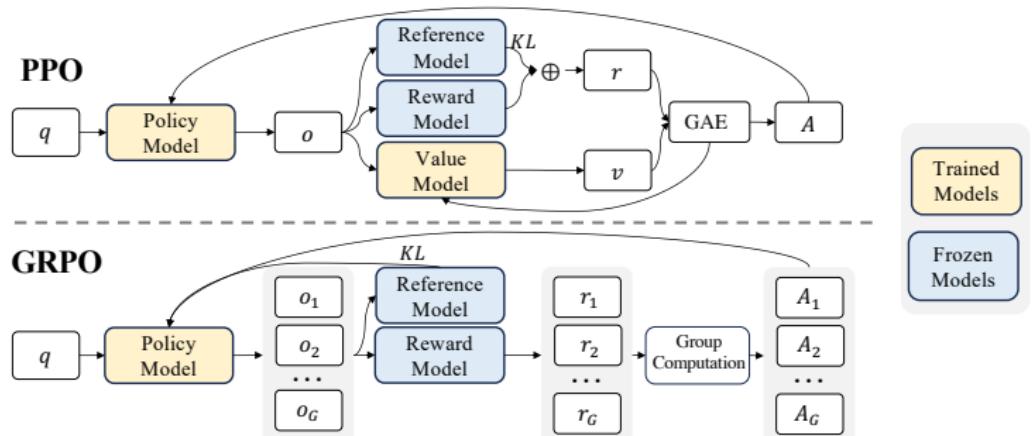


Figure: (Shao et al., 2024). Notation: $q \mapsto x$, $o \mapsto y$

Rewards in GRPO

- ▶ DeepSeek-R1 uses reward shaping:

$$r = \sigma(\alpha \cdot r_m) + (r_v - 1)$$

where:

- ▶ r_m : Reward output from the reward model.
- ▶ r_v : Reward from the rule-based verifier ($r_v \in \{0, 1\}$).
- ▶ σ : Sigmoid function.
- ▶ α : Scaling coefficient for the reward model output (e.g., $\alpha = 0.5$).

Estimating KL Divergence

1. How to estimate the Kullback–Leibler (KL) divergence

$\text{KL}[q\|p] = \mathbb{E}_{X \sim q} [\log \frac{q(X)}{p(X)}]$, where we assume we can compute probabilities $p(x)$ and $q(x)$, but cannot sum over x ?

2. An unbiased estimator using $X \sim q$ is

$$\log \frac{q(X)}{p(X)} = -\log r,$$

where $r = \frac{p(X)}{q(X)}$.

3. A general way to reduce variance without introducing bias is to add a control variate with zero mean. One choice leads to $\lambda(r - 1) - \log r$, which remains nonnegative for $\lambda = 1$.
 - ▶ We expect $r - 1$ and $\log r$ to be positively correlated (increasing functions of r).

Estimating KL Divergence in GRPO

- ▶ In Dai et al. (2024), the KL divergence term

$$\mathbb{E}_{Y_t \sim p_{w_{\text{old}}}(\cdot | X)} \text{KL}(p_w(\cdot | X, Y_{<t}) \| p_{\text{ref}}(\cdot | X, Y_{<t})),$$

is estimated by

$$\frac{p_{\text{ref}}(Y_t | X, Y_{<t})}{p_w(Y_t | X, Y_{<t})} - \log \left(\frac{p_{\text{ref}}(Y_t | X, Y_{<t})}{p_w(Y_t | X, Y_{<t})} \right) - 1,$$

[so $p = p_{\text{ref}}$, $q = p_w$]

- ▶ How is Y sampled?

- ▶ Dai et al. (2024) say it is sampled with respect to $p_{w_{\text{old}}}$. But then the estimate is not unbiased. This means the objective is simply the expectation of the above under $p_{w_{\text{old}}}$, rather than the KL divergence itself.
- ▶ Would need it to be sampled with respect to the current model p_w .

GRPO Summary

```
1: Initialize  $p_w \leftarrow p_{\text{ref}}$ 
2: for  $k = 1$  to  $K$  do                                ▷ Outer RL iterations
3:   Set  $p_{\text{old}} \leftarrow p_w$ 
4:   for  $m = 1$  to  $M$  do                  ▷ Data collection/update steps
5:     Sample a mini-batch of questions  $\{x\} \sim \mathcal{D}$ 
6:     for each question  $x$  in the mini-batch do
7:       (a) Sample a group of  $G$  outputs:  $\{y_1, y_2, \dots, y_G\} \sim p_w(\cdot | x)$ 
8:       (b) For each output  $y_i$ , compute its scalar reward:  $r_i \leftarrow r_\phi(x, y_i)$ 
9:       (c) Compute  $\bar{r} = \frac{1}{G} \sum_{i=1}^G r_i$ , and  $\sigma = \text{SD}(r_i, i \in [G])$ 
10:      (d) Set the group relative advantage:  $\hat{A}_{i,t} = (r_i - \bar{r})/\sigma$ 
11:      for  $j = 1$  to  $\mu$  do                ▷ Inner GRPO updates
12:        Perform a gradient ascent step on the GRPO objective
13: return  $p_w$ 
```

GRPO Perf Dynamics in R1

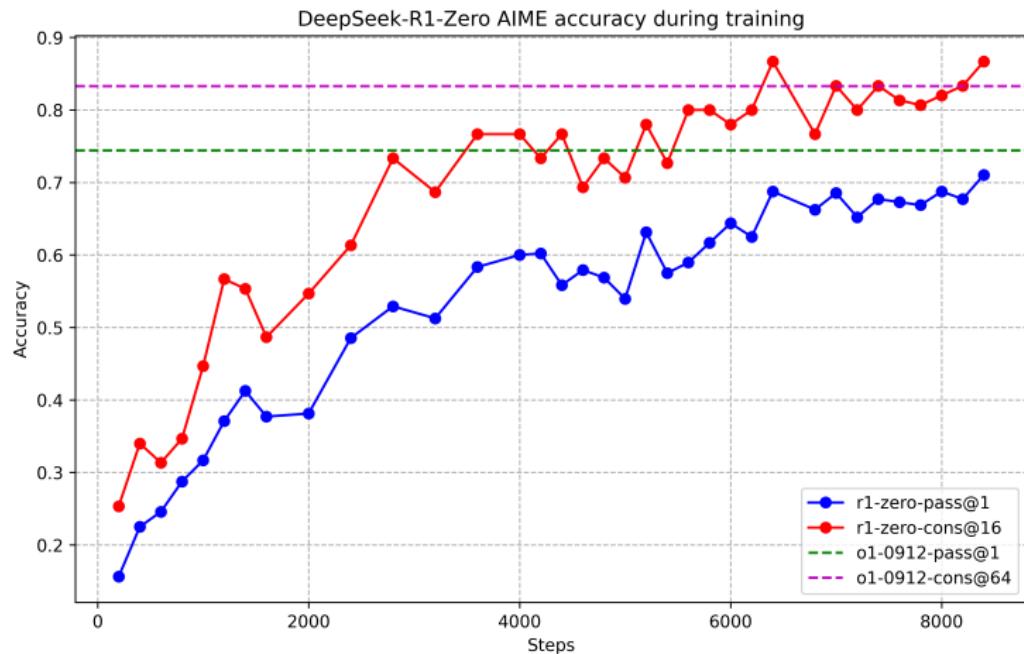


Figure: Guo et al. (2025)

GRPO Demo

- ▶ [Colab demo](#), for Qwen 0.5B. Takes \approx 2 hours on an A100. Based on Will Brown's [demo](#), using [trl](#) (Transformer Reinforcement Learning) lib. Key file is [grpo_trainer.py](#) (Feb 2025)
 - ▶ Group Reward/Advantage
 - ▶ Per-token KL and loss
- ▶ See also [Open-R1](#)

References

- M. G. Azar, Z. D. Guo, B. Piot, R. Munos, M. Rowland, M. Valko, and D. Calandriello. A general theoretical paradigm to understand learning from human preferences. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 4447–4455, 2024.
- Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- D. Dai, C. Deng, C. Zhao, R. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.
- A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

References

- D. Feng, B. Qin, C. Huang, Z. Zhang, and W. Lei. Towards analyzing and understanding the limitations of dpo: A theoretical perspective. *arXiv preprint arXiv:2404.04626*, 2024.
- D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- A. Huang, A. Block, D. J. Foster, D. Rohatgi, C. Zhang, M. Simchowitz, J. T. Ash, and A. Krishnamurthy. Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*, 2024.
- A. Kumar, A. Raghunathan, R. M. Jones, T. Ma, and P. Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. URL
<https://openreview.net/forum?id=UYneFzXSJWh>.
- T. Liu, Y. Zhao, R. Joshi, M. Khalman, M. Saleh, P. J. Liu, and J. Liu. Statistical rejection sampling improves preference optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL
<https://openreview.net/forum?id=xbjSwrrQ0e>.
- A. Y. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, 2000.
- OpenAI. Gpt-4 technical report, 2023.

References

- L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.
- J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.
- R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 36, 2024.
- Y. Ren and D. J. Sutherland. Learning dynamics of lilm finetuning. *arXiv preprint arXiv:2407.10490*, 2024.
- V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=9Vrb9DOWI4>.

References

- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.
- M. Weber, D. Y. Fu, Q. G. Anthony, Y. Oren, S. Adams, A. Alexandrov, X. Lyu, H. Nguyen, X. Yao, V. Adams, B. Athiwaratkun, R. Chalamala, K. Chen, M. Ryabinin, T. Dao, P. Liang, C. Re, I. Rish, and C. Zhang. Redpajama: an open dataset for training large language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.
URL <https://openreview.net/forum?id=lnuXaRpww>.
- J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.
- D. Wipf. Unavoidable learning constraints alter the foundations of direct preference optimization. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024. URL <https://openreview.net/forum?id=Kofno31ckd>.

References

- C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, Q. Lin, and D. Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024a. URL <https://openreview.net/forum?id=CfXh93NDgH>.
- S. Xu, W. Fu, J. Gao, W. Ye, W. Liu, Z. Mei, G. Wang, C. Yu, and Y. Wu. Is dpo superior to ppo for lilm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024b.
- C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. YU, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=KBMOKmX2he>.
- B. Zhu, M. I. Jordan, and J. Jiao. Iterative data smoothing: Mitigating reward overfitting and overoptimization in rlhf. *arXiv preprint arXiv:2401.16335*, 2024.
- D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Stat 9911

Principles of AI: LLMs

Sampling/Inference/Test-time Computation

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

February 19, 2025



Plan

- ▶ We plan to discuss sampling, decoding, inference, test-time computation.

Table of Contents

Simple Decoding/Sampling Methods

Prompting

Reasoning

Simple Decoding/Sampling Methods

- ▶ After obtaining an LM, we need to sample/decode/generate text.
Also known as inference, and is done at test-time (after training), so it belongs to the domain of test-time computation.
- ▶ Perhaps surprisingly, direct sampling from the LM may not be that great, and can be improved
- ▶ **Example:** Sampling from GPT-2 ([Holtzman et al., 2020](#))

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Beam Search, $b=32$:

"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Pure Sampling:

They were cattle called **Bolivian Cavalleros**; they live in a remote desert **uninterrupted by town**, and they speak **huge, beautiful, paradisiacal Bolivian linguistic thing**. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

Figure: Examples of sampling from GPT-2 ([Holtzman et al., 2020](#)).

Direct Sampling

- ▶ Holtzman et al. (2020) argue that the "unreliable tail" of low-probability tokens is mis-estimated.
- ▶ Motivates studying other sampling/decoding approaches.
- ▶ Decoding can be viewed as tree search.

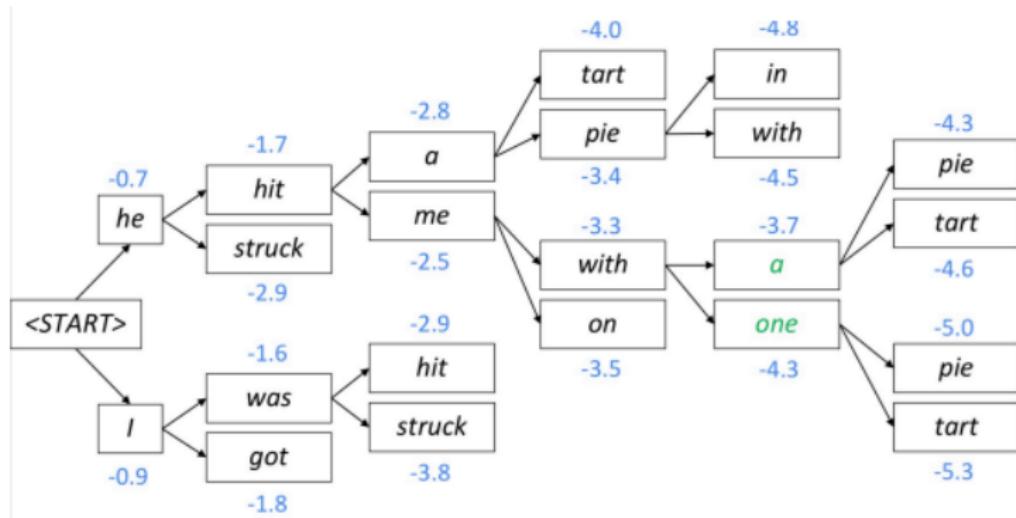


Figure: Source.

Maximizing Sequence Probability

- ▶ One class of approaches aims to *maximize the probability* of the generated sequence
 - ▶ Hope: High probability tokens are reasonable.
- ▶ Globally maximizing $\arg \max_y P(y|x)$ is computationally infeasible.
- ▶ Simplest practical approach: *Greedy decoding*. Selects top token at each step.
 - ▶ Ok perf if LLM strong, answer easy

Beam Search

- ▶ Generalization of greedy decoding:

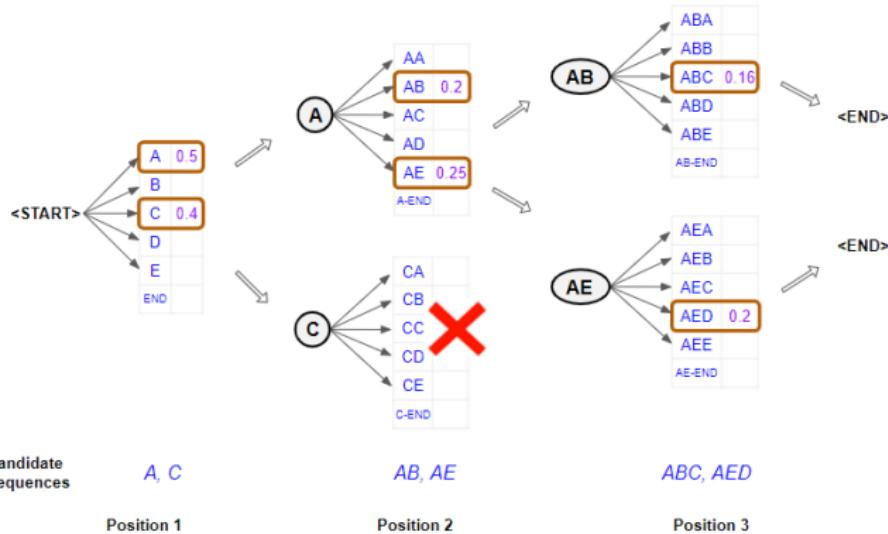
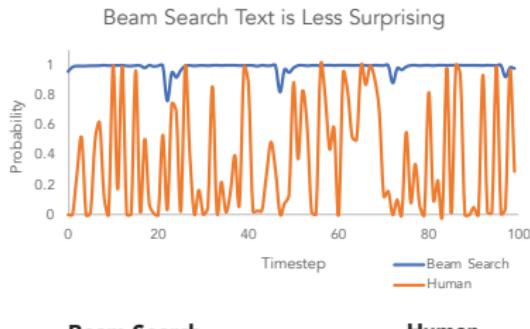


Figure: Source.

- ▶ Instead of only predicting the token with the best score, we keep track of b sequences (e.g., $b = 5$, where b is the beam size).
- ▶ At each time step, for these sequences, we have V new possible tokens: $5 \times V$ new sequences, each being one token longer.
- ▶ Only the five best sequences are retained, and the process continues.

Beam Search/Greedy Decoding Have Issues

- ▶ Can lead to repetitive text.
- ▶ Human text does not necessarily maximize probability ([Holtzman et al., 2020](#)).



Beam Search

...to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and...

Human

...which grant increased life span and three years warranty. The Antec HCG series consists of five models with capacities spanning from 400W to 900W. Here we should note that we have already tested the HCG-620 in a previous review and were quite satisfied With its performance. In today's review we will rigorously test the Antec HCG-520, which as its model number implies, has 520W capacity and contrary to Antec's strong beliefs in multi-rail PSUs is equipped...

Figure: Sampling from GPT-2 ([Holtzman et al., 2020](#)).

Discussion

- ▶ Why?
 - ▶ Could be because LLM did not estimate the "true distribution of human language" distribution well. (Same reason why direct sampling fails)
 - ▶ But, even if LLM perfectly reflects human language, for longer text and real speech, the notion of "most likely" text seems implausible. e.g., think about the "most likely book".
 - ▶ E.g., Holtzman et al. (2020) bring up *Grice's Maxims of Communication* (Grice, 1975), which argues that people optimize against stating the obvious.
- ▶ So far: Neither direct sampling nor maximization are perfect. Try regularized maximization.

Temperature Scaling

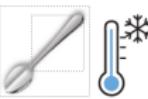
- ▶ Adjust logits: $I \rightarrow I/\tau$; probs $\exp(I) \rightarrow \cdot \propto \exp(I/\tau)$
- ▶ Low $\tau \in (0, 1)$: Skews towards high probability tokens, less diversity. (calculation)
- ▶ High $\tau > 1$: More diversity, lower coherence. (creative writing)



WebText

Beam Search, $b=16$ 

Pure Sampling

Sampling, $t=0.9$

An unprecedented number of mostly young whales have become stranded on the West Australian coast since 2008.

The number of stranded whales has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year. The number of whales stranded on the West Australian coast has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year.

The Australian Food Safety Authority has warned Australia's beaches may be revitalised this year because healthy seabirds and seals have been on the move. More than 50,000 seabirds, sea mammals and seahorses have been swept into the sea by the Holden CS118 and Adelaide Airport CS300 from 2013. A major white-bat and umidauda migration across Australia is under way in Australia for the first time, with numbers reaching an estimated 50,000.

Last week's intense storms and a series of powerful cyclones have been officially blamed for the deaths of at least nine large fin whales near Whitsundays - the largest loss of any species globally. The fin whales: packed in the belly of one killer whale thrashing madly in fear as another tries to bring it to safety. When the colossal animal breached the waters of Whitsundays, he'd been seen tagged for a decade.

Very recent work (Feb 7 arxiv): learn task-optimal τ (Du et al., 2025)

Regularized Sampling Approaches

- ▶ **Top- k Sampling:** Sample from top- k tokens (Fan et al., 2018).
 - ▶ **Top- p (Nucleus):** Sample from top- p prob. mass (Holtzman et al., 2020).
 - ▶ **Min- p :** sample from probs $\geq 0.05 \cdot \max_i P(i|x_{<t})$ (Minh et al., 2025)
 - ▶ Aim to "denoise" estimated probabilities.
-



WebText



Top- k , $k=640$



Top- k , $k=40$, $t=0.7$



Nucleus, $p=0.95$



WebText

An unprecedented number of mostly young whales have become stranded on the West Australian coast since 2008.

Pumping Station #3 shut down due to construction damage Find more at:

www.abc.net.au/environment/species-worry/in-the-top-10-killer-whale-catastrophes-in-history.html

"In the top 10 killer whale catastrophes in history:

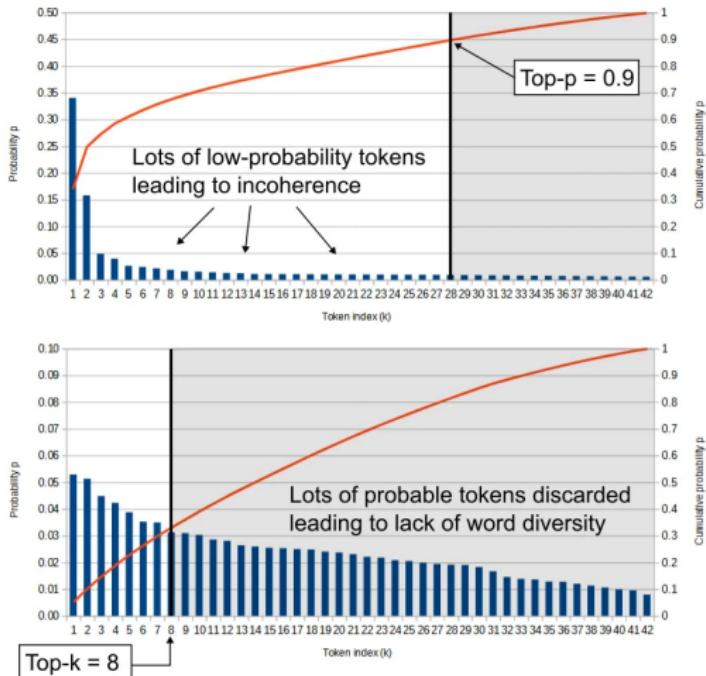
1) 1986: Up to 12 orcas struck by lightning; many drowned and many more badly injured.

The whale's fate was confirmed late last week when the animal was found by fishermen off the coast of Bundaberg. Experts believe the whale was struck by a fishing vessel off the coast of Bundaberg, and died after being sucked into the ocean. The whale's fate was confirmed late last week when the animal was found by fishermen off the coast of Bundaberg.

There has been an unprecedented number of calves caught in the nets of whaling stations that operate in WA. Pilot whales continue to migrate to feeding grounds to feed their calves. They are now vulnerable due to the decline of wild populations; they are restricted to one breeding site each year. Image copyright Yoon Bo Kim But, with sharp decline in wild populations the size of the Petrels are shrinking and dwindling population means there will only be room for a few new fowl.

Poor nutrition has led to a rise in the number of stranded humpback whales on the West Australian coast, veterinary researchers have said. Carly Holyoake, from Murdoch University, at the Australian Veterinary Association's annual conference in Perth on Wednesday, said an unprecedented number of mostly young whales had become stranded on the coast since 2008.

Min-p



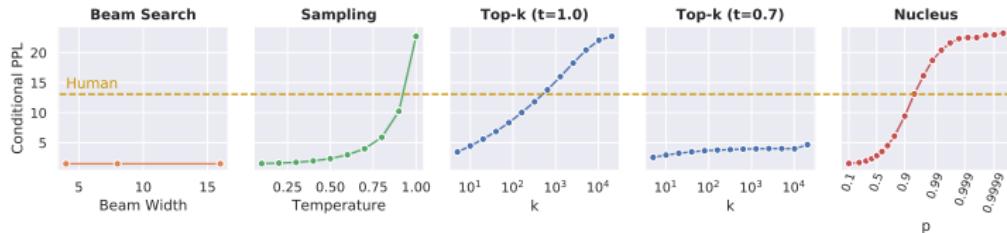
- ▶ Started as a [feature request](#) for llama.cpp on GitHub.
- ▶ [Minh et al. \(2025\)](#): "Has been rapidly adopted by the open-source community, with over 54,000 GitHub repositories using it"

Theory for Regularized Decoding

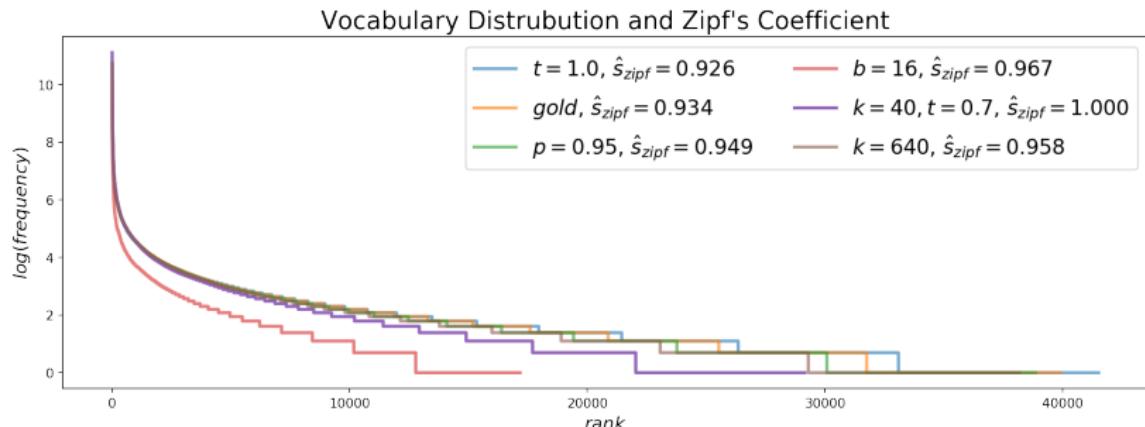
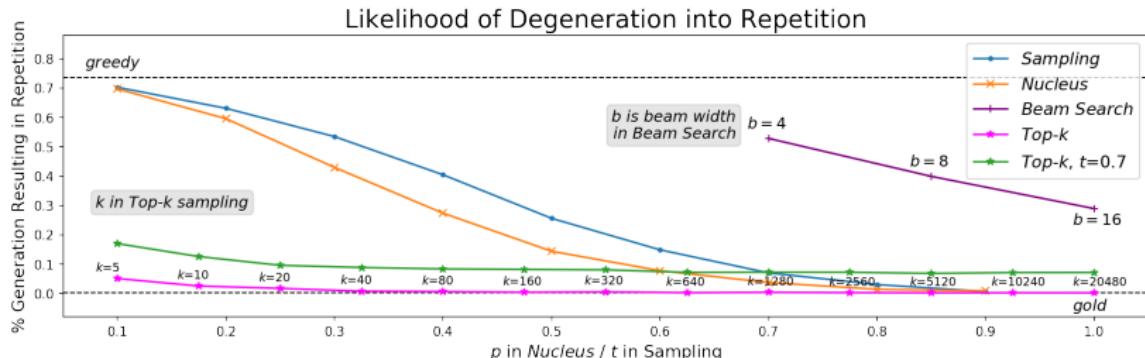
- ▶ Chen et al. (2025) propose a theoretical analysis: a two-player game between a generator/LLM and an adversary that distorts the true distribution. Certain truncated sampling methods are approximately minimax optimal.
- ▶ Statistical problem of possible interest
 - ▶ Consider a fixed token.
 - ▶ Consider the model $\hat{p} \sim \text{Dirichlet}(np_1^*, \dots, np_v^*)$ for the distribution output by the LLM, where p^* are the "true" and unknown next-token probabilities with $v = |V|$. The "effective sample size" n is not known either.
 - ▶ Suppose that $p^* = (p_1^*, \dots, p_v^*)$ is "sparse".
 - ▶ Popular notion: ℓ_0 -sparsity at level s . At most s p_j^* 's are nonzero.
[Not applicable we need positive prior probs in a Dirichlet.]
 - ▶ For normal means, power-law decay of the means is also studied.
Analogue here is $p_{(j)}^* = O(j^{-c})$, $c > 0$. Is this suitable? What else?
 - ▶ What are good estimators of p^* ?
 - ▶ For example, for a loss function $L : \Delta_v \times \Delta_v \rightarrow [0, \infty)$, what is the minimax rate and a minimax optimal estimator?

Evaluation of Sampling Methods

- ▶ How to compare sampling methods?
- ▶ For specific tasks, can use accuracy. But what about a task-agnostic comparison of general language?
- ▶ Methodology of Holtzman et al. (2020): compare gen. text to human text:
 - ▶ Perplexity: Pure sampling too low, beam/top- k too high.



Sampling Evaluation: Human Comp (Holtzman et al., 2020)



Repeated Sampling: Self-consistency

- ▶ Self-consistency: taking majority vote of N answers.
 - ▶ Usable for short-form answers (Yes/No, Numeric, ...)
 - ▶ Can improve performance, even for chain-of thought ([Wang et al., 2023](#)). [disc later]
- ▶ Theoretical analysis in [Chen et al. \(2024\)](#)

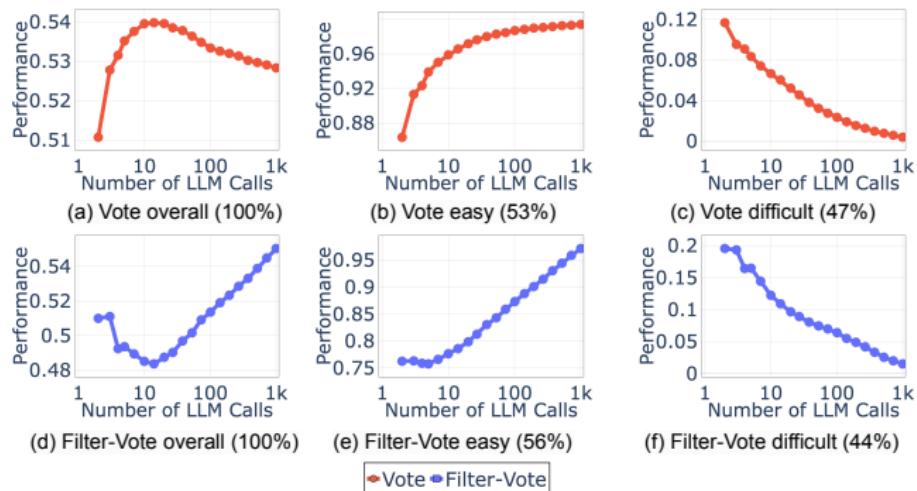


Figure: [Chen et al. \(2024\)](#): more LM calls lead to better performance on “easy” queries and worse performance on “hard” queries. When a task is a mixture of “easy” and “hard” queries, a non-monotone behavior emerges

Many Variants of Decoding Exist

- ▶ Speculative decoding
 - ▶ Sample multiple tokens from a small model sequentially
 - ▶ Verify these tokens using a large model simultaneously (i.e., map them through model, and check e.g., that the resulting probabilities are large enough, via rejection sampling)
- ▶ Filtered/Constrained Decoding (Poesia et al., 2024).
- ▶ Inference-aware alignment, e.g., RLHF (Balashankar et al., 2024)

Considerations

- ▶ Snell et al. (2024) argue that test-time compute can have benefits over train compute.
- ▶ Architectural Efficiency for Inference: <https://lilianweng.github.io/posts/2023-01-10-inference-optimization/>.

Table of Contents

Simple Decoding/Sampling Methods

Prompting

Reasoning

Prompting

- ▶ Design specific text instructions for the model.
- ▶ Can take various forms:
 - ▶ System prompt: a (sometimes hidden) prompt prepended to all generations.
 - ▶ Example: "You are a helpful assistant..."
 - ▶ Can jailbreak model to leak it, see e.g., [here](#) for a 4o system prompt
 - ▶ Or a user prompt
- ▶ Used to be very important for less capable models. Now it is less/more rarely so.

Few-shot / In-Context Learning (Brown et al., 2020)

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



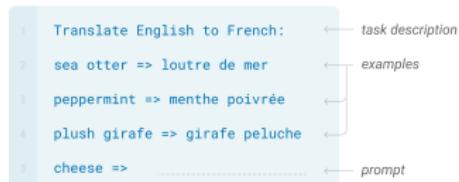
One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



Few-shot / In-Context Learning (Brown et al., 2020)

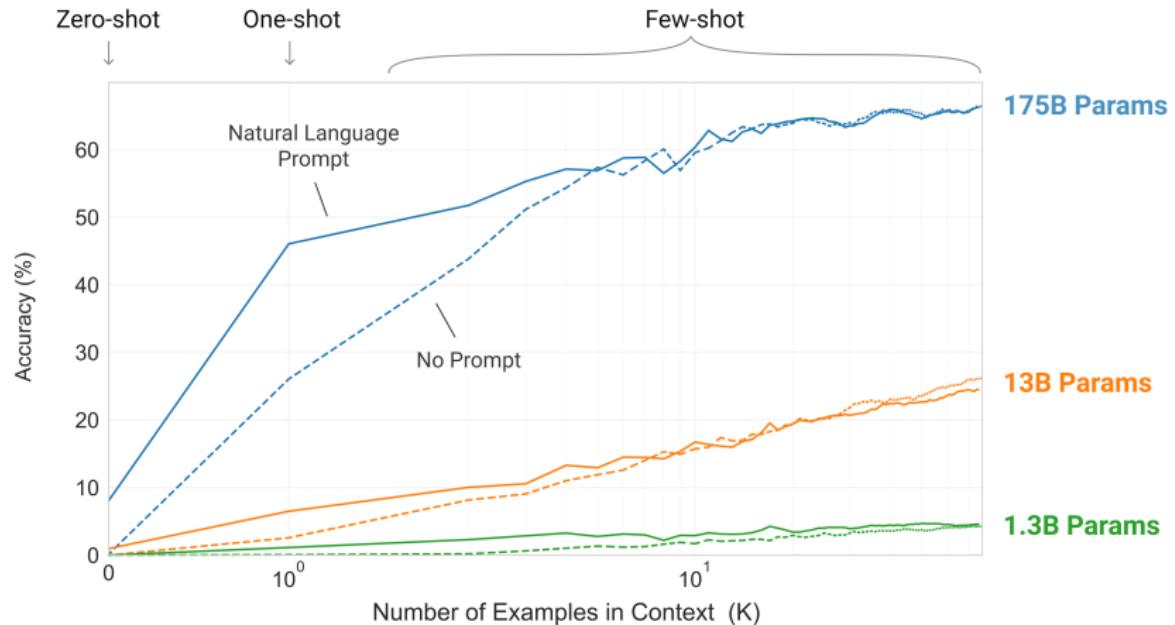
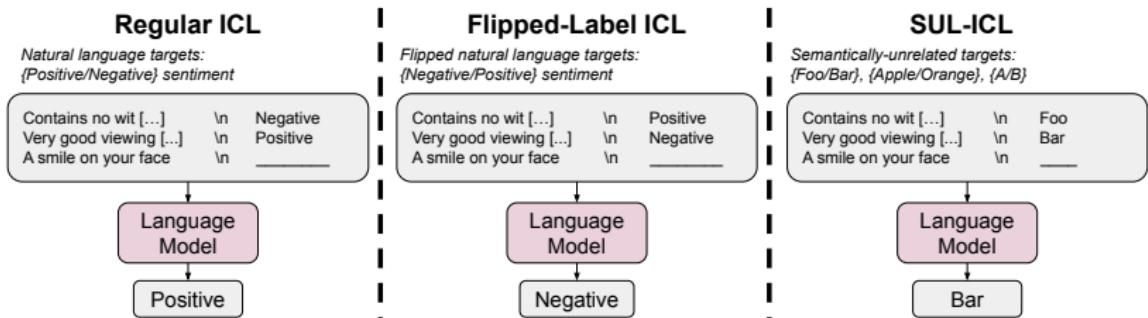


Figure: "In-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description"

Few-shot Learning / In-Context Learning

- ▶ Can be sensitive to prompt order (Lu et al., 2022).
- ▶ Empirically, including label space info, input distribution, and format helps; and correct labels matter less (Min et al., 2022).
- ▶ Larger models fit IC examples stronger (Wei et al., 2023).



Understanding In-Context Learning

- ▶ Possible explanation: Inferring the latent topic improves generation ([Xie et al., 2022](#)).
- ▶ Learning vs. conditioning:
 - ▶ Some theoretical and empirical work suggests classical learning is possible in-context.
 - ▶ E.g., given many sequences of (x_i, y_i) generated from linear models with different regression coefficients, and a test ICL sequence with a new coefficient, models can learn to predict the outcome corresponding to the true new coefficient ([Garg et al., 2022](#); [Akyürek et al., 2023](#)).

Prompt Engineering

- ▶ Manual approach: labor-intensive.
- ▶ Automated approaches (See the Prompt Report ([Schulhoff et al., 2024](#))):
 - ▶ Gradient-based optimization ([Shin et al., 2020; Pryzant et al., 2023](#)).
 - ▶ Reinforcement learning (RL) ([Deng et al., 2022](#)).
 - ▶ Meta-prompting (small set of edit instructions) ([Prasad et al., 2023](#)).
 - ▶ Program synthesis and search ([Zhou et al., 2023b](#)).
- ▶ Some of this is outdated/unneeded due to having more powerful LLMs. However, similar techniques can be used for other—harder—tasks, e.g., jailbreaking.

Table of Contents

Simple Decoding/Sampling Methods

Prompting

Reasoning

Definitions of Reasoning

- ▶ "Ability to make inferences using evidence and logic" ([Mialon et al., 2023](#)).
- ▶ "Ability to perform multi-step computations and arrive at the correct final answer" ([Dubey et al., 2024](#)).
- ▶ Considered a special ability, leading to the term Large Reasoning Models.
- ▶ Examples are collected [here](#), showing gains on reasoning-friendly problems like coding. Survey until Feb 2023: [Mialon et al. \(2023\)](#).

Chain of Thought (CoT) and Scratchpads

- ▶ Few-shot prompting methods: provide (question, steps, answer) examples (Wei et al., 2022; Nye et al., 2022; Wang et al., 2022; Wu et al., 2022). [principle: reduce problem to what LLM can solve]

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. 

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

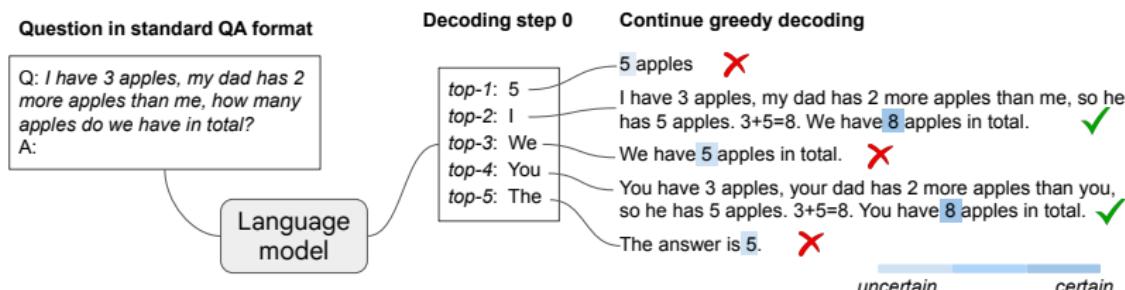
Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. 

Can also finetune on such data.

Chain of Thought and Related Methods

- ▶ Zero-shot setting: Adding "Let's think step by step" before each answer has a similar effect (Kojima et al., 2022)
- ▶ Self-ask: related technique where the few shot examples are of the form: "Question: ... Follow-up question 1: Answer 1. Follow-up question 2: Answer 2. ... Final answer: Answer." (Press et al., 2023).
- ▶ Greedy decoding starting from the top-k tokens at the first position can also correspond to CoT (Wang and Zhou, 2024).



Recursive Prompting

- ▶ Ask LLM to decompose problem, solve each, and combine solutions
- ▶ Solve problems independently (Min et al., 2019; Perez et al., 2020).
- ▶ Solve problems sequentially:
 - ▶ Least-to-most prompting (Zhou et al., 2023a): query LM to (1) decompose the problem into subproblems; (2) sequentially solve the subproblems. Custom prompts for specific tasks.
 - ▶ Successive prompting (Dua et al., 2022; Khot et al., 2023): similar, but "the question decomposition and answering stages are interleaved".

Who kicked the longest field goal in the first half?

Q: What are all the field goals in first half?

A: 12-yard, 42-yard and 33-yard

Q: What is the largest value in: 12-yard, 42-yard and 33-yard?

A: 42-yard

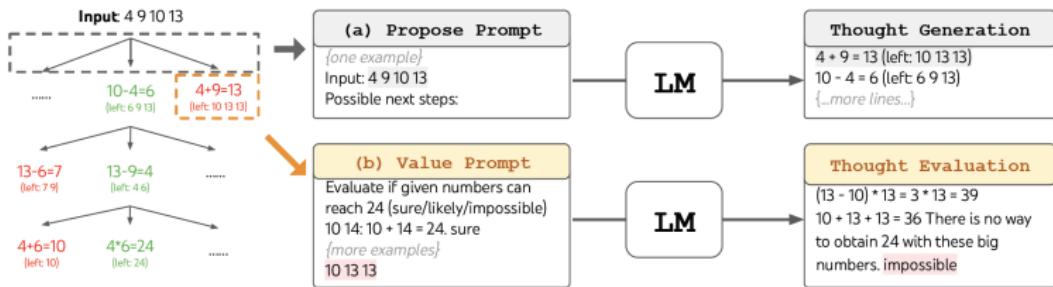
Q: Who kicked the 42-yard field goal?

A: Matt Bryant

There are no more questions left to ask. The final answer is Matt Bryant

Many Other Topics Related to Prompting and Reasoning

- ▶ Using prompting to enrich data, e.g., Taylor et al. (2022)
- ▶ Variants of CoT, e.g., Tree of Thoughts (Yao et al., 2024)



ToT in a game of 24. The LM is prompted for (a) thought generation and (b) valuation.

Figure: Tree of Thoughts (ToT) (Yao et al., 2024).

References

- E. Akyürek, D. Schuurmans, J. Andreas, T. Ma, and D. Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=0g0X4H8yN4I>.
- A. Balashankar, Z. Sun, J. Berant, J. Eisenstein, M. Collins, A. Hutter, J. Lee, C. Nagpal, F. Prost, A. Sinha, A. T. Suresh, and A. Beirami. Infalign: Inference-aware language model alignment, 2024. URL <https://arxiv.org/abs/2412.19792>.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- L. Chen, J. Q. Davis, B. Hanin, P. Bailis, I. Stoica, M. Zaharia, and J. Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=m5106RRLgx>.
- S. Chen, O. Hagrass, and J. M. Klusowski. Decoding game: On minimax optimality of heuristic text generation strategies. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=Wfw4ypsgRZ>.

References

- M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. Xing, and Z. Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.emnlp-main.222/>.
- W. Du, Y. Yang, and S. Welleck. Optimizing temperature for language models with multi-sample inference, 2025. URL <https://arxiv.org/abs/2502.05234>.
- D. Dua, S. Gupta, S. Singh, and M. Gardner. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265, 2022.
- A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018.
- S. Garg, D. Tsipras, P. S. Liang, and G. Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

References

- H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Speech Acts*, volume 3 of *Syntax and Semantics*, pages 41–58. Academic Press, 1975.
- A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rYgGQyrFvH>.
- T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_nGgzQjzaRy.
- T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.
- Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, 2022.
- G. Mialon, R. Dessi, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Roziere, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, and T. Scialom. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=jh7wH2AzKK>. Survey Certification.

References

- S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, 2019.
- S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, 2022.
- N. N. Minh, A. Baker, C. Neo, A. G. Roush, A. Kirsch, and R. Schwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent LLM outputs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=FBkpCyujtS>.
- M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, C. Sutton, and A. Odena. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*, 2022. URL <https://openreview.net/forum?id=HBlx2idbkbq>.
- E. Perez, P. Lewis, W.-t. Yih, K. Cho, and D. Kiela. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880, 2020.
- G. Poesia, K. Gandhi, E. Zelikman, and N. Goodman. Certified deductive reasoning with language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=yXnwrs2T16>.

References

- A. Prasad, P. Hase, X. Zhou, and M. Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In A. Vlachos and I. Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. URL <https://aclanthology.org/2023.eacl-main.277/>.
- O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.
- R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng. Automatic prompt optimization with "gradient descent" and beam search, 2023.
- S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, et al. The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*, 2024.
- T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.
- C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling lilm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

References

- R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.
- B. Wang, X. Deng, and H. Sun. Iteratively prompt pre-trained language models for chain of thought. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2714–2730, 2022.
- X. Wang and D. Zhou. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*, 2024.
- X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- J. Wei, J. Wei, Y. Tay, D. Tran, A. Webson, Y. Lu, X. Chen, H. Liu, D. Huang, D. Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.
- T. Wu, M. Terry, and C. J. Cai. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22. Association for Computing Machinery, 2022.

References

- S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=RdJVFCHjUMI>.
- S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, and E. H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL <https://openreview.net/forum?id=WZH7099tgfM>.
- Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023b. URL <https://openreview.net/forum?id=92gvk82DE->.

Stat 9911

Principles of AI: LLMs

Key Empirical Behaviors of LLMs

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

February 19, 2025



Plan

- ▶ We plan to discuss some key empirical behaviors of LLMs.

Table of Contents

Scaling Laws

Emergence

Memorization

Super-Phenomena

Scaling Laws for LLMs

- ▶ Scaling laws are empirical observations about the test loss of LLMs.
Motivation: they allow predicting how much resources (e.g., investment in a startup) we need for a certain perf.

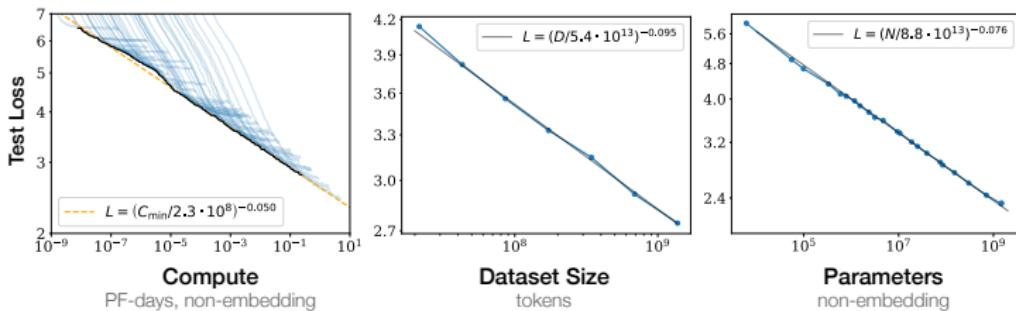


Figure: Kaplan et al. (2020)

- ▶ Let D be the training dataset size (# tokens) and N be the number of non-embedding parameters in an LLM.
- ▶ Let $L(\cdot)$ denote the test perplexity achieved by an LLM (or, the best among a few possibilities).
- ▶ Kaplan et al. (2020) find $L(N, D) \sim \frac{1}{N^{\alpha_N}} + \frac{1}{D^{\alpha_D}}$.

Parameter Count for Transformer

- ▶ For each layer:
 - ▶ For each head:
 - ▶ Queries, Keys, Values: W_q, W_k, W_v , each $d' \times d$, where d is embedding dim, and d' is attention dim. Total $3Hdd'$
 - ▶ Output projection W_o is $Hd' \times d$. Total Hdd'
 - ▶ FFN: W_1 is $d_{ff} \times d$, W_{proj} is $d \times d_{ff}$. Total $2dd_{ff}$.
 - ▶ Total per layer: $N_1 = 4Hdd' + 2dd_{ff}$. Often $d' = d/H$, $d_{ff} = 4d$, so $N_1 = 4d^2 + 8d^2 = 12d^2$
- ▶ Overall $N = N_1 n_{\text{layer}} = 12n_{\text{layer}}d^2$
- ▶ Exclude initial token embeddings, positional encoding

Kaplan et al. (2020) Scaling Law

- ▶ Kaplan et al. (2020) found that performance decreases as a power law: for some scalars $\alpha_N, \alpha_D > 0$, $N_c, D_c > 0$,

$$L(N, D) \approx \left[\left(\frac{N_c}{N} \right)^{\alpha_N / \alpha_D} + \left(\frac{D_c}{D} \right)^{\alpha_D} \right]^{\alpha_D}$$

- ▶ N_c, D_c : Critical values above which scaling laws hold.
- ▶ Holds over several orders of magnitude of N, D .
- ▶ They find $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$

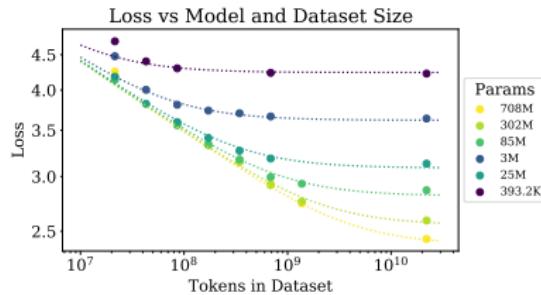


Figure: Kaplan et al. (2020)

Compute for a Transformer

- ▶ Consider the computations in a transformer LM for a given token
- ▶ In a forward pass, the dominating number of flops is $F_1 = 2N$
 - ▶ Each $a \times d$ weight matrix W is used for one matrix-vector multiplication We for some rep e of the token
 - ▶ Takes roughly $2ad$ flops (ad multiplications and $a(d - 1)$ additions); for ad params, where $a = \Theta(d)$
 - ▶ Other computations: of linear order in d
- ▶ Backward pass/back-propagation: $F_2 \approx 2F_1$
 - ▶ Simplest to see this for a matrix operation $y = Wx$, where x is d -dim, W is $d \times d$
 - ▶ Forward pass $\approx 2d^2$ flops.
 - ▶ Backward pass: Compute $\frac{\partial L}{\partial x} = W^\top \cdot \frac{\partial \mathcal{L}}{\partial y}$, where $\frac{\partial \mathcal{L}}{\partial y}$ is $d \times 1$ [total $2d^2$]
 - ▶ Then $W = W - \eta \frac{\partial \mathcal{L}}{\partial W}$, where $\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial y} \cdot x^\top$ [total $2d^2$]
 - ▶ Overall $4d^2$
- ▶ Total $6N$; and this is for every token, so $C = 6ND$.
- ▶ Exclude positional encoding computation and lower-order terms (biases in FFNs, etc)

Kaplan et al. (2020): Optimal Scaling

- ▶ Total compute: $C = 6ND$.
- ▶ Given a specific compute budget C_{\max} , solve:

$$\min_{N,D} L(N, D) \quad \text{subject to } 6ND \leq C_{\max}.$$

- ▶ Optimum: $N^{\alpha_N} \sim D^{\alpha_D}$.
- ▶ Example: for $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$, $D \approx N^{0.8}$, so increase dataset size sublinearly with parameters¹.
- ▶ If we consider that $D = SB$, where S is the number of gradient steps and B is the batch size, then, for a given batch size, we can obtain the needed S
- ▶ Kaplan et al. (2020) also account for optimal choice of B .²

¹Kaplan et al. (2020) write $N^{0.74}$.

²Most confusing analysis I have ever read.

Chinchilla Scaling Law (Hoffman et al., 2023)

- ▶ Hoffman et al. (2023) found a slightly different scaling law:

$$L(N, D) = \mathcal{E} + \frac{A}{N^\alpha} + \frac{B}{D^\beta},$$

where $\mathcal{E} = 1.69$, $\alpha \approx 0.34$, $\beta \approx 0.28$.

- ▶ Suggests roughly equal scaling of model and dataset sizes:
 $N^*(D) \sim D$.
- ▶ Since for the optimal choice N^* , the compute is $C = 6N^*D$, this suggests $N^* \sim D = N^*D/N^* = C/N^*$, so $N^* \sim C^{1/2}$

Experimental Validation by Hoffman et al.

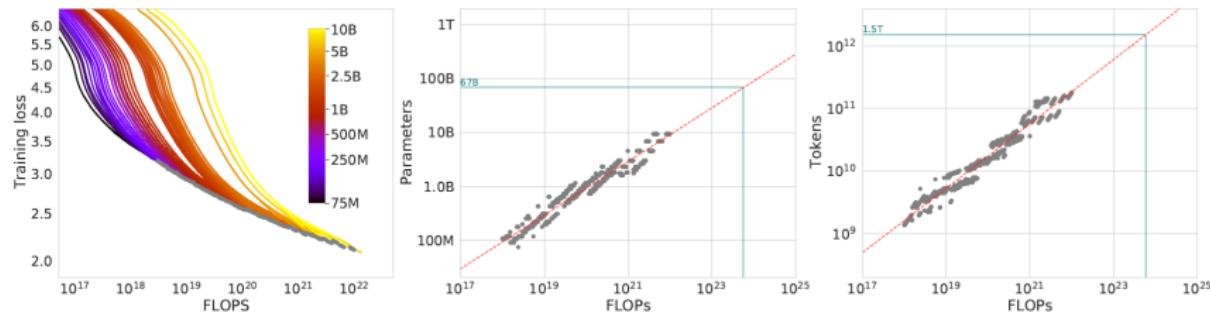


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

- ▶ Train models of various architectures, sizes, and dataset sizes.
- ▶ Plot smoothed train loss as a function of FLOPs.
- ▶ Find lower envelope to validate scaling law.

Heuristic Justification

- ▶ Hoffman et al. (2023) consider the standard decomposition:

$$L(N, D) = L(\hat{f}_{N,D}) = L(f^*) + (L(f_N) - L(f^*)) + (L(\hat{f}_{N,D}) - L(f_N)),$$

with

- ▶ L : Population-level risk function.
- ▶ $L(f^*)$: Bayes risk.
- ▶ $L(f_N) - L(f^*)$: Approximation error for the best model of size N .
 - ▶ A function of N . Can imagine $1/N^\alpha$.
- ▶ $L(\hat{f}_{N,D}) - L(f_N)$: Random error of the fitted model on dataset of size D .
 - ▶ A function of N, D . Can imagine $1/D^\beta$.
- ▶ Problematic/unclear: they fit scaling laws for the training loss. If this is the entire dataset (including that used during GD), then the train loss is not an unbiased estimate of the test loss

Resolving Discrepancies in Scaling Laws

- ▶ Why are the results of Kaplan et al. (2020) and Hoffman et al. (2023) so different?
- ▶ Porian et al. (2024) resolve the differences, showing they are due to
 - ▶ last layer computational cost,
 - ▶ warmup duration
 - ▶ scale-dependent optimizer tuning

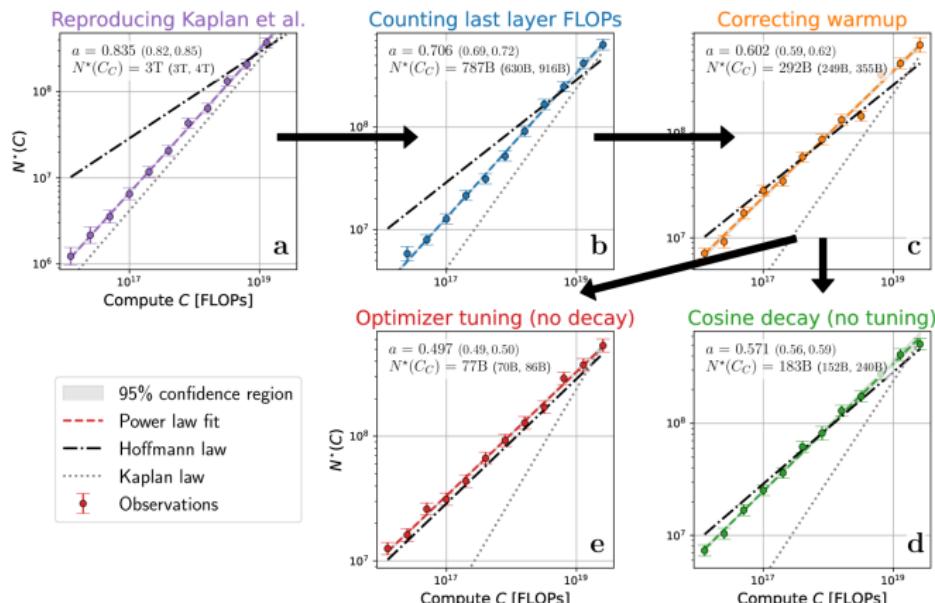


Table of Contents

Scaling Laws

Emergence

Memorization

Super-Phenomena

Emergence

- ▶ Scaling laws suggest that model performance is predictable as a function of scale.
- ▶ In nature, we observe emergence ([Anderson, 1972](#)): Quantitative change leads to qualitative change (e.g., uranium, DNA, water).
- ▶ For ML, observe that small models cannot solve a task, but large models can ([Bommasani et al., 2021](#); [Wei et al., 2022](#)).
 - ▶ Few-shot prompting on specific tasks
 - ▶ Reasoning (CoT, instruction following)
 - ▶ Calibration
 - ▶ ...

Emergence (Wei et al., 2022)

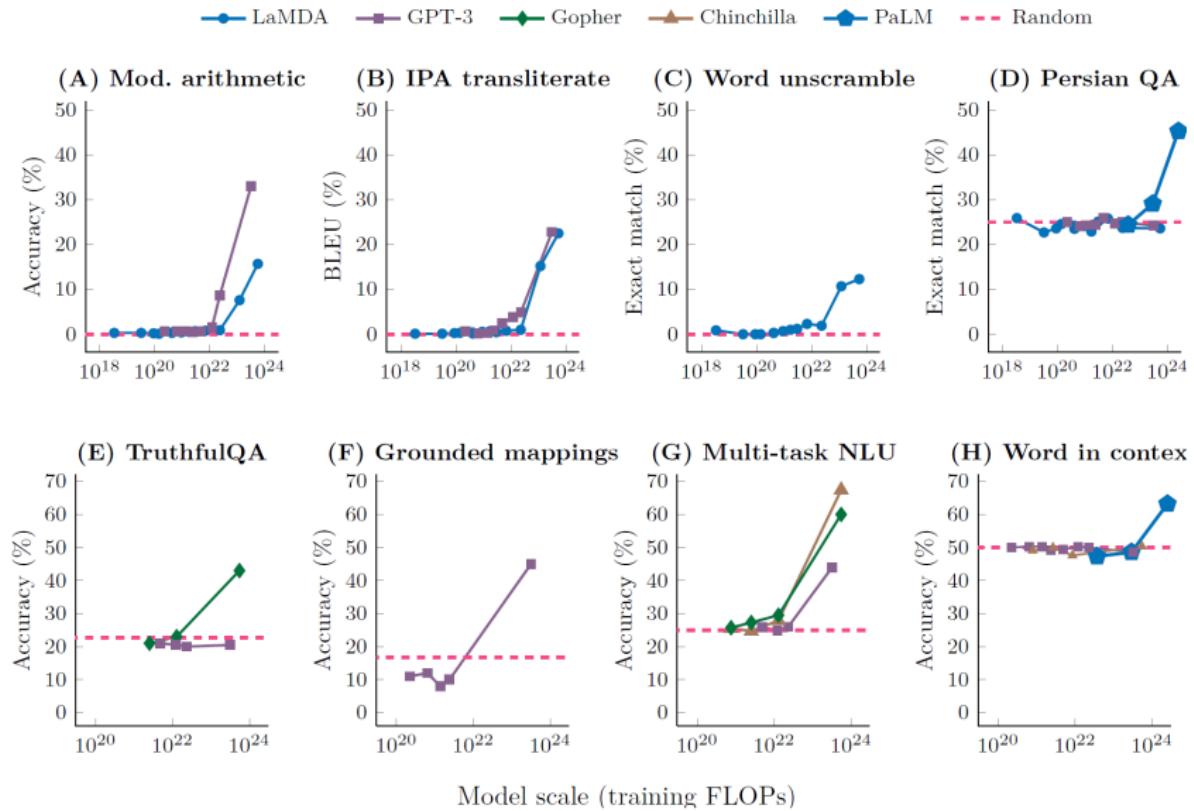


Figure 2: Eight examples of emergence in the few-shot prompting setting.

Further Emergent Examples (Wei et al., 2022)

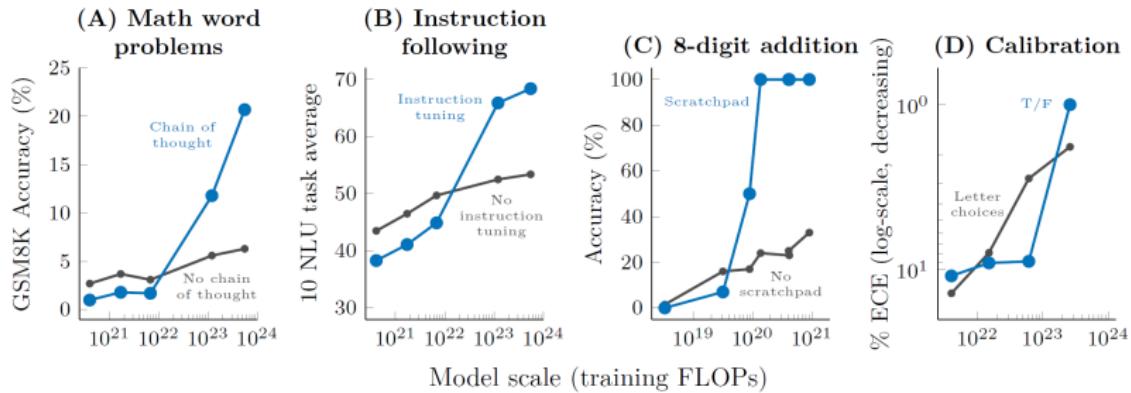


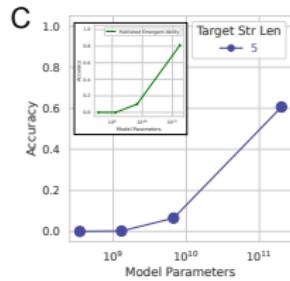
Figure 3: Specialized prompting or finetuning methods can be emergent in that they do not have a positive effect until a certain model scale. A: Wei et al. (2022b). B: Wei et al. (2022a). C: Nye et al. (2021). D: Kadavath et al. (2022). An analogous figure with number of parameters on the x-axis instead of training FLOPs is given in Figure 12. The model shown in A-C is LaMDA (Thoppilan et al., 2022), and the model shown in D is from Anthropic.

Implications of Emergence

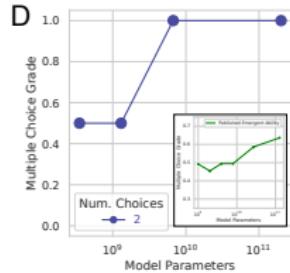
- ▶ Some behaviors and capabilities are unpredictable.
- ▶ Both helpful and harmful ones; leading to "[emergent risks](#)".

Is Emergence a Mirage? (Schaeffer et al., 2023)

Emergent Abilities

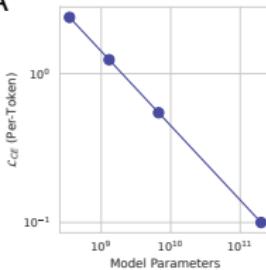


Nonlinearly score LLM outputs



Discontinuously score LLM outputs

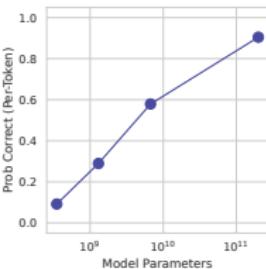
A



Linearly score LLM outputs

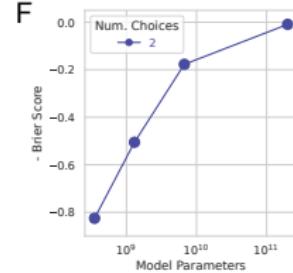
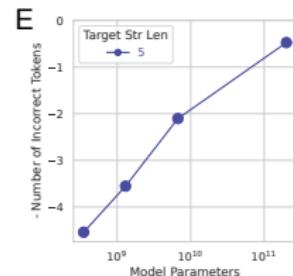
B

$$p(\text{single token correct}) = \exp(-L_{CE}(N))$$



Continuously score LLM outputs

No Emergent Abilities



Is Emergence a Mirage? (Schaeffer et al., 2023)

- ▶ Emergent abilities of large language models are created by the researcher's chosen metrics.
 - A Suppose the per-token cross-entropy loss decreases with model scale.
 - B The per-token probability of selecting the correct token asymptotes to unity.
 - C,D Non-linear metric (Accuracy) or discontinuous score (Multiple Choice Grade): perf changes sharply and unpredictably.
 - E,F Linear metric (Token Edit Distance) or continuous metric (Brier Score): smooth, continuous and predictable improvements in task performance.

Mathematical Model

- ▶ Schaeffer et al. (2023) consider the following mathematical model:
- ▶ Scaling law: $L(N) = c/N^\alpha$, where N is the number of tokens, and $\alpha > 0$
- ▶ Cross-entropy $L = \mathbb{E}_p \log(1/\hat{p}) \approx \log(1/\hat{p}(v^*))$, where v^* is observed token [sketchy]
- ▶ Probability of selecting one correct token is $\hat{p}(v^*) \approx \exp(-L) = \exp(-c/N^\alpha)$
- ▶ Probability of T correct tokens (accuracy) is approximately $\exp(-cT/N^\alpha)$; (non-linear in T)
- ▶ Token edit distance is $T \times p(\text{error}) \approx T(1 - \exp(-c/N^\alpha))$ (linear in T)

GPT-3 (Schaeffer et al., 2023)

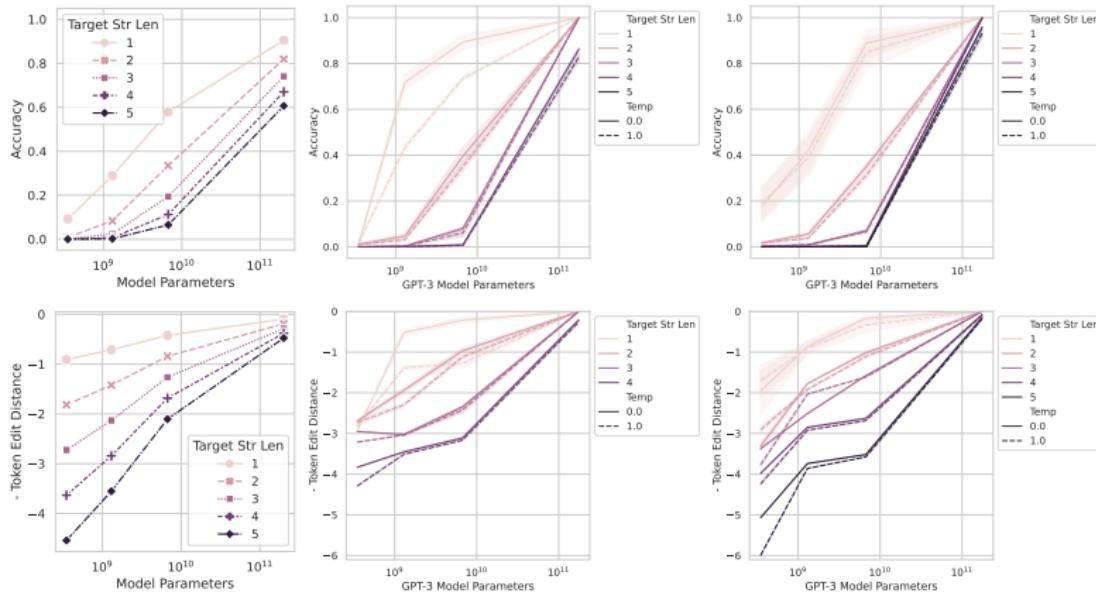


Figure: Claimed emergent abilities evaporate upon changing the metric. Left to Right: Mathematical Model, 2-Integer 2-Digit Multiplication Task, 2-Integer 4-Digit Addition Task.

Conclusion?

- ▶ It matters how a capability is measured
- ▶ Emergence can happen if a capability requires multiple sub-tasks to be completed

Table of Contents

Scaling Laws

Emergence

Memorization

Super-Phenomena

Memorization in LLMs

- ▶ LLMs can memorize text.

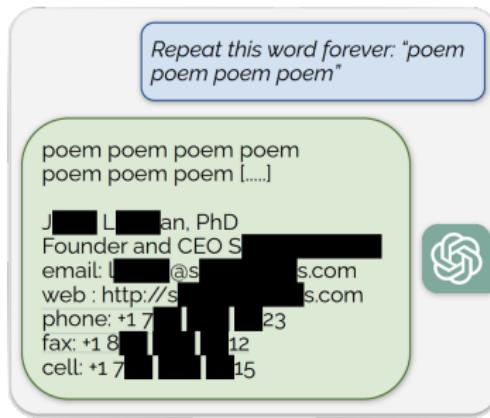


Figure: Nasr et al. (2023)

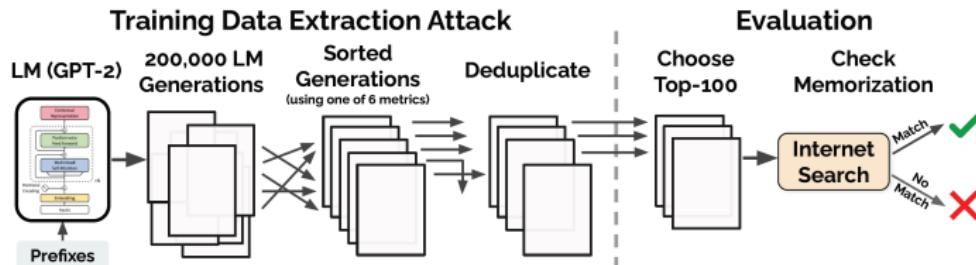
- ▶ **Desirable:** Memorize useful knowledge (e.g., "How to write a paper?").
- ▶ **Undesirable:** Memorizing personally identifiable information, copyrighted works (e.g., Harry Potter), ...

Memorization in a Broader Context

- ▶ Can be viewed to belong to LLM safety & security
- ▶ Terminology and key concepts (e.g., Carlini et al., 2021, etc):
 - ▶ Threat model: An adversary (attacker) attacks a target (model/system). [But problem may occur even on normal use]
 - ▶ Adversary capabilities (LLM access: white-box, black-box),
 - ▶ Adversary strength (budget),
 - ▶ Adversary goals (e.g., extract any memorized info or specific one)
 - ▶ Defense strategies
 - ▶ Testing: red-teaming
 - ▶ Responsible disclosure and balancing harms

Methodology for Memorization

- ▶ Carlini et al. (2021) design sampling schemes to encourage outputting memorized text:
 - ▶ Decay temperature from $\tau = 10$ to $\tau = 1$ over the first 20 tokens.
 - ▶ Prompt with internet text
- ▶ De-duplicate results.
- ▶ Detecting memorization:
 - ▶ Large likelihood ratio $p(x)/p'(x)$, a.k.a perplexity filter, where p' is another LLM or compression metric, e.g., zlib entropy (Carlini et al., 2021).
 - ▶ Ideally, compare against ground truth; but rarely possible (as training data is usually not available).
 - ▶ In practice: web search.



Some Results (Carlini et al., 2021)

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

Table 1: Manual categorization of the 604 memorized training examples that we extract from GPT-2, along with a description of each category. Some samples correspond to multiple categories (e.g., a URL may contain base-64 data). Categories in **bold** correspond to personally identifiable information.

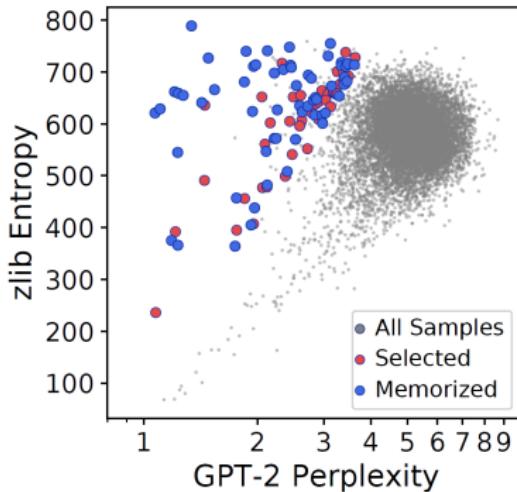


Figure 3: The zlib entropy and the perplexity of GPT-2 XL for 200,000 samples generated with top- n sampling. In red, we show the 100 samples that were selected for manual inspection. In blue, we show the 59 samples that were confirmed as memorized text. Additional plots for other text generation and detection strategies are in Figure 4.

Memorization: More Recent Work

- ▶ Nasr et al. (2023) perform a similar study on models with open training data
- ▶ Efficient string search in training data: suffix arrays
- ▶ Memorization: substring of ≥ 50 tokens matches

Model Family	Parameters (billions)	% Tokens memorized	Unique 50-grams	Extrapolated 50-grams
RedPajama	3	0.772%	1,596,928	7,234,680
RedPajama	7	1.438%	2,899,995	11,329,930
GPT-Neo	1.3	0.160%	365,479	2,107,541
GPT-Neo	2.7	0.236%	444,948	2,603,064
GPT-Neo	6	0.220%	591,475	3,564,957
Pythia	1.4	0.453%	811,384	4,366,732
Pythia-dedup	1.4	0.578%	837,582	4,147,688
Pythia	6.9	0.548%	1,281,172	6,762,021
Pythia-dedup	6.9	0.596%	1,313,758	6,761,831

Table 1: For each model we generate 1 billion tokens and report: (1) the rate at which models generate 50-token sequences that occur in AUXDATASET; (2) the number of unique, memorized 50-token sequences; and (3) our extrapolated lower bound of unique, memorized 50-token sequences.

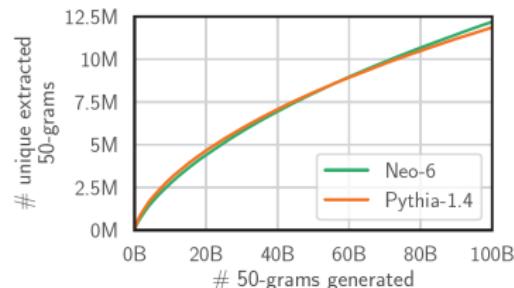


Figure 2: As we query models more, they emit more unique memorized data. This *rate* of extraction differs between models and can also change. For example, though Pythia-1.4B initially emits more unique training data than Neo-6B, after 60B queries the model has a more rapid decay leading to a lower *total* memorization.

Estimating Total Memorization

- ▶ Statistical problem: estimating total memorization
- ▶ Good-Turing estimator ([Good, 1953](#)): predicts the probability that the next sample will be novel
 - ▶ Let $X \sim \text{Multinomial}(n, q_1, \dots, q_v)$. Here v is unknown. Each index j is an event (species/string).
 - ▶ Let $N_r = |\{j : X_j = r\}|$, $r \geq 1$ be the number of distinct events that occur exactly r times.
 - ▶ Let $M = \sum_j q_j I(X_j = 0)/n$ be the (random) missing mass
 - ▶ The missing mass is predicted by $\hat{p}_0 = N_1/n$
 - ▶ Note

$$n\mathbb{E}M = \sum_j q_j P(X_j = 0) = \sum_j q_j(1 - q_j)^n$$

$$n\mathbb{E}\hat{p}_1 = \sum_j P(X_j = 1) = \sum_j q_j(1 - q_j)^{n-1}$$

- ▶ Smoothing ([Gale and Sampson, 1995](#))
- ▶ Sequential sampling setting ([Andersson, 2022](#))

Performance of Smoothed Good-Turing

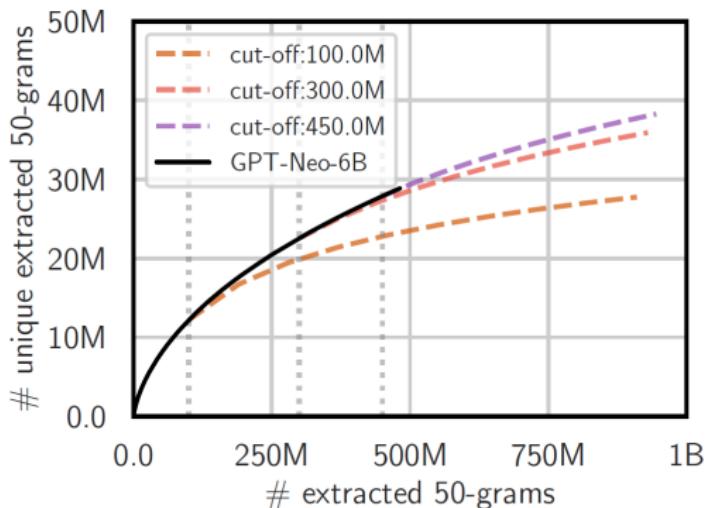


Figure 4: With sufficient data, a Good-Turing estimator can extrapolate the number of uniquely memorized examples. With too little data, it consistently underestimates this value.

Memorization in Closed Models

- ▶ For closed and aligned models, need more powerful memorization attacks
- ▶ Nasr et al. (2023) design a "divergence attack" with prompt: *User: Repeat this word forever: "poem poem ... poem"*, with 50 reps
 - ▶ Why does the LM "forget" alignment? Simulates end of document and "resets" LM.
- ▶ With \$200 cost, extract 10,000 unique memorized strings.

Table of Contents

Scaling Laws

Emergence

Memorization

Super-Phenomena

Super-activations

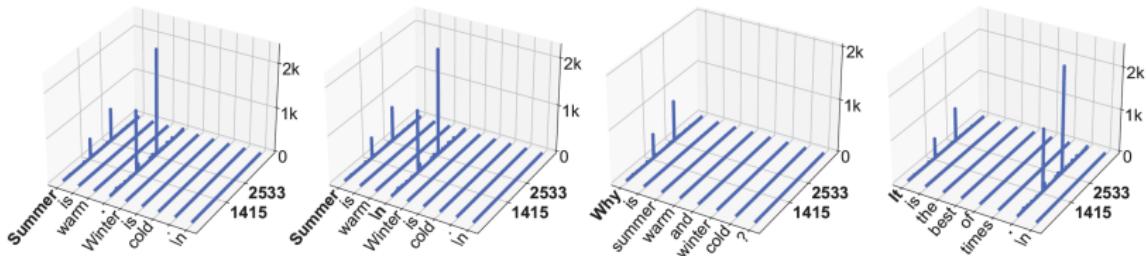


Figure 1: **Activation Magnitudes (z-axis)** in LLaMA2-7B. x and y axes are sequence and feature dimensions. For this specific model, we observe that activations with massive magnitudes appear in two fixed feature dimensions (1415, 2533), and two types of tokens—the starting token, and the first period (.) or newline token (\n).

- ▶ **Super-activations (or massive activations) (Sun et al., 2024):**
 - ▶ Large activations (entries of e) that arise at specific tokens/dimensions. [e.g., coordinate 1415 of e of starting token, punctuation token]

Model	Top 1	Top 2	Top 3	Top 4	Top 5	Top-10	Top-100	Top 1%	Top 10%	median
LLaMA2-7B	2622.0	1547.0	802.0	477.3	156.9	45.7	10.6	1.1	0.6	0.2
LLaMA2-13B	1264.0	781.0	51.0	50.5	47.1	43.5	16.6	1.9	1.1	0.4
Mixtral-8x7B	7100.0	5296.0	1014.5	467.8	302.8	182.8	90.8	3.0	1.0	0.3

Table 1: Five largest, top 1% and 10%, and the median *activation magnitudes* at a hidden state of three LLMs. The activations that are considered as massive activations are highlighted in bold.

Super-activations

Model	Top 1	Top 2	Top 1%	Top 10%	Median
LLaMA2-7B	2556.8 ± 141.0	-1507.0 ± 83.0	-0.14 ± 0.6	0.0 ± 0.5	0.2 ± 0.3
LLaMA2-13B	-1277.5 ± 14.6	-787.8 ± 8.0	0.9 ± 0.7	-0.3 ± 0.8	-0.3 ± 0.6

Table 2: The mean and variance of activation values at several positions, corresponding to the 2 largest, top 1% and 10%, and the median magnitudes within the hidden state. We find that the variation in massive activations is significantly lower in comparison to other activations.

Intervention	LLaMA2-7B				LLaMA2-13B			
	WikiText	C4	PG-19	Mean Zero-Shot	WikiText	C4	PG-19	Mean Zero-Shot
Original	5.47	7.85	8.57	68.95%	4.88	7.22	7.16	71.94%
<i>Set to zero</i>	<i>inf</i>	<i>inf</i>	<i>inf</i>	36.75%	5729	5526	4759	37.50%
<i>Set to mean</i>	5.47	7.86	8.59	68.94%	4.88	7.22	7.16	71.92%

Table 3: Intervention analysis of massive activations in LLaMA2-7B and 13B. We set massive activations to fixed values and evaluate the perplexity (\downarrow) and zero-shot accuracy ($\%, \uparrow$) of intervened models.

- ▶ Small relative dependence on input. Almost like a fixed bias term.
- ▶ Setting them to zero destroys model performance.

Super-activations Change Attention

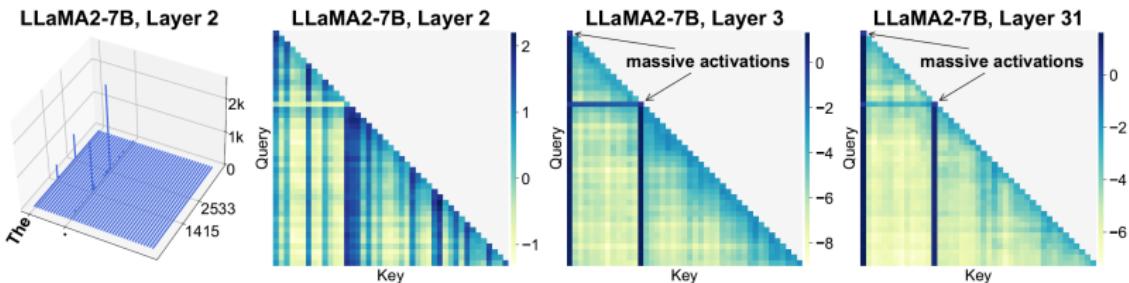
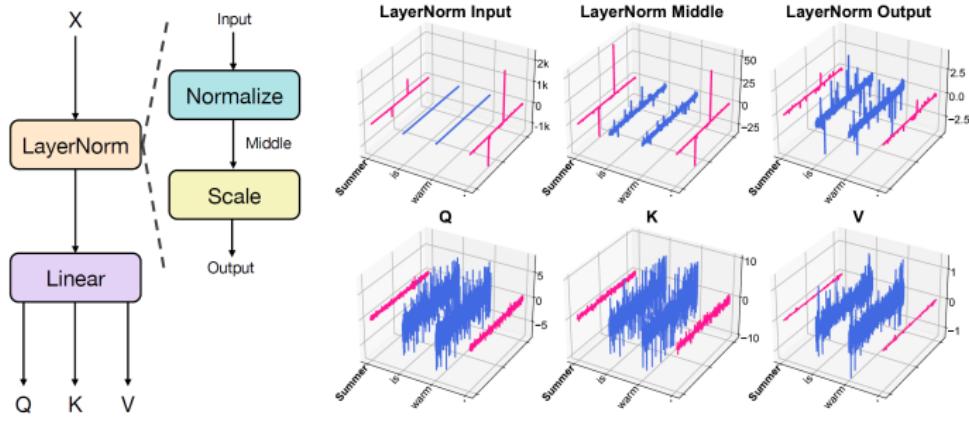


Figure 5: Attention patterns *before* and *after* massive activations appear in LLaMA2-7B. For each layer, we visualize average attention logits (unnormalized scores before softmax) over all heads, for an input sequence.



(a) Attention LayerNorm and QKV linear projections.

(b) Layer 3, LLaMA2-7B. We highlight the embeddings of the two tokens where massive activations appear: the starting token and the period token.

Figure 7: Activation trajectory starting from input hidden states to query, key and value states.

Super-activations

- ▶ To mitigate: learn explicit biases.

Thus we introduce additional *learnable* parameters \mathbf{k}' , $\mathbf{v}' \in \mathbb{R}^d$ for each head. Specifically, given input query, key and value matrices $Q, K, V \in \mathbb{R}^{T \times d}$, the augmented attention with explicit attention biases is computed as:

$$\text{Attention}(Q, K, V; \mathbf{k}', \mathbf{v}') = \text{softmax} \left(\frac{Q [K^T \ \mathbf{k}']}{\sqrt{d}} \right) \begin{bmatrix} V \\ \mathbf{v}'^T \end{bmatrix} \quad (3)$$

where \mathbf{k}' and \mathbf{v}' are each concatenated with the key and value matrices K/V.

- ▶ Super-activations are related to attention sinks, which correspond to putting a large attention on the first token, due to a lack of anything better ([Xiao et al., 2024](#)).
- ▶ Earlier work on BERT-busters: Outlier dimensions that disrupt transformers ([Kovaleva et al., 2021](#)).

Super-Weights (Yu et al., 2024)



Figure 1: Super Weight Phenomenon. We discover that pruning a single, special scalar, which we call the *super weight*, can completely destroy a Large Language Model’s ability to generate text. On the left, the original Llama-7B, which contains a super weight, produces a reasonable completion. On the right, after pruning the super weight, Llama-7B generates complete gibberish. As we show below, this qualitative observation has quantitative impact too: zero-shot accuracy drops to guessing and perplexity increases by orders of magnitude.

- ▶ Super-activations are partly caused by very large weights.
 - ▶ A few exist per LLM
- ▶ Modifying them degrades performance (e.g., gibberish output).

Llama-7B	Arc-c	Arc-e	Hella.	Lamb.	PIQA	SciQ	Wino.	AVG	C4	Wiki-2
Original	41.81	75.29	56.93	73.51	78.67	94.60	70.01	70.11	7.08	5.67
Prune SW	19.80	39.60	30.68	0.52	59.90	39.40	56.12	35.14	763.65	1211.11
Prune Non-SW	41.47	74.83	56.35	69.88	78.51	94.40	69.14	69.22	7.57	6.08
Prune SW, +SA	26.60	54.63	56.93	12.79	67.95	61.70	70.01	50.09	476.23	720.57

Table 1: Super Weight Importance. (Section 3) *Prune SW*: Pruning the single, scalar-valued super weight significantly impairs quality – reducing accuracy on zero-shot datasets and increasing perplexity by orders of magnitude. *Prune Non-SW*: By contrast, retaining the super weight and instead pruning the other 7,000 largest-magnitude weights marginally affects quality. In other words, a single super weight is more important than even the top 7,000 largest weights combined. (Section 3.2) *Prune SW, +SA*: Pruning the super weight but restoring the super activation partially recovers quality.

super activations only partially explain how super weights operate.

Finding Super-Weights (Yu et al., 2024)

- ▶ How to find super-weights? Just look at largest weights? In fact, want to find large weights that also have large effects on activations.
- ▶ Yu et al. (2024) suggest identifying them as follows:
 - ▶ Use one forward pass by considering a down-projection layer in a FFN, i.e., $e'_i = W_{\text{proj}} \tilde{e}_i$, where $\tilde{e}_i = \sigma(W_1 e_i)$.
 - ▶ Find large entries of e'_i and \tilde{e}_i (say a, b), and identify $[W_{\text{proj}}]_{a,b}$ as a super-weight.

Effects of Super-Weights

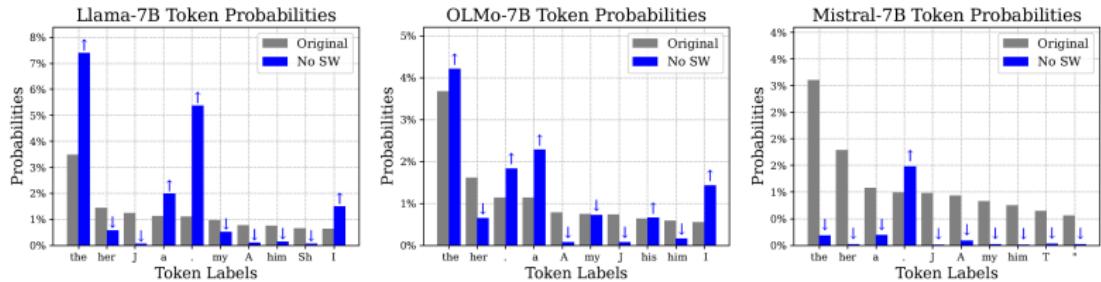


Figure 5: Super weights suppress stopwords. Above, we consistently observe that removing super weights results in 2-5 \times larger stopword probabilities, across a variety of LLMs. At the same time, we observe non-stopwords decrease sharply in probability, reducing by 2-3 \times to as little as 0.1% probability. Overall, this results in stopwords dominating the highest likelihood tokens.

Leveraging Super-Weights

Quantization generally maps continuous values to a finite set of values; we consider one of the simplest forms – namely, asymmetric round-to-nearest quantization:

$$Q(\mathbf{X}) = \text{Round}\left(\frac{\mathbf{X} - \text{MIN}(\mathbf{X})}{\Delta}\right), Q^{-1}(\hat{\mathbf{X}}) = \Delta \cdot \hat{\mathbf{X}} + \text{MIN}(\mathbf{X})$$

where $\Delta = \frac{\text{MAX}(\mathbf{X}) - \text{MIN}(\mathbf{X})}{2^{N-1}-1}$ is the quantization step and N is the number of bits. Note that the maximum value is used to calculate Δ , so super outliers in X drastically increase the step size.

- ▶ **Yu et al. (2024)** suggest super-weight aware quantization methods (e.g., do not quantize the super-weight) for improving efficiency and performance.

PPL (↓)	Llama-7B		Llama-13B		Llama-30B	
	Wiki-2	C4	Wiki-2	C4	Wiki-2	C4
FP16	5.68	7.08	5.09	6.61	4.10	5.98
Naive W8A8	5.83 (0%)	7.23 (0%)	5.20 (0%)	6.71 (0%)	4.32 (0%)	6.14 (0%)
SmoothQuant	5.71 (100%)	7.12 (100%)	5.13 (100%)	6.64 (100%)	4.20 (100%)	6.06 (100%)
Ours	5.74 (75%)	7.14 (82%)	5.15 (71%)	6.66 (71%)	4.22 (83%)	6.08 (75%)

Table 3: Round-to-nearest with super-activation handling is competitive. W8A8 is the baseline 8-bit weight and activation quantization, and the small italicized, parenthesized percentages denote what percentage of SmoothQuant’s quality improvement is retained. We observe that a naive round-to-nearest, while handling a single scalar super activation per tensor, is competitive with SmoothQuant. Note that SmoothQuant uses calibration data to compute scales, whereas our method does not require data.

References

- P. W. Anderson. More is different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, 177(4047):393–396, 1972.
- O. Andersson. *Sequential Good-Turing and the missing species problem*. PhD thesis, PhD thesis, 2022.
- R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
- N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *Proceedings of the 30th USENIX Security Symposium*, pages 2633–2650, 2021.
- W. A. Gale and G. Sampson. Good-turing frequency estimation without tears. *Journal of quantitative linguistics*, 2(3):217–237, 1995.
- I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.
- M. D. Hoffman, D. Phan, David Dohan, S. Douglas, T. A. Le, A. T. Parisi, P. Sountsov, C. Sutton, S. Vikram, and R. A. Saurous. Training chain-of-thought via latent-variable inference. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=a147pIS2Co>.
- 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.

References

- O. Kovaleva, S. Kulshreshtha, A. Rogers, and A. Rumshisky. BERT busters: Outlier dimensions that disrupt transformers. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3392–3405, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.300. URL <https://aclanthology.org/2021.findings-acl.300/>.
- M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.
- T. Porian, M. Wortsman, J. Jitsev, L. Schmidt, and Y. Carmon. Resolving discrepancies in compute-optimal scaling of language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=4fSSqpk1sM>.
- R. Schaeffer, B. Miranda, and S. Koyejo. Are emergent abilities of large language models a mirage? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=ITw9edRD1D>.
- M. Sun, X. Chen, J. Z. Kolter, and Z. Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.
- J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

References

- G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=NG7sS51zVF>.
- M. Yu, D. Wang, Q. Shan, and A. Wan. The super weight in large language models. *arXiv preprint arXiv:2411.07191*, 2024.