

LLM Scaling & Factual Knowledge

Slido: <https://app.sli.do/event/t5K64Ct2uzLSEPbAUtrqLU>

Yu Meng

University of Virginia
yumeng5@virginia.edu

Oct 27, 2025

Overview of Course Contents

- Week 1: Logistics & Overview
- Week 2: N-gram Language Models
- Week 3: Word Senses, Semantics & Classic Word Representations
- Week 4: Word Embeddings
- Week 5: Sequence Modeling & Recurrent Neural Networks (RNNs)
- Week 6: Language Modeling with Transformers
- Week 8: Transformer and Pretraining
- Week 9: Large Language Models (LLMs) & In-context Learning
- **Week 10: Knowledge in LLMs and Retrieval-Augmented Generation (RAG)**
- Week 11: LLM Alignment
- Week 12: Reinforcement Learning for LLM Post-Training
- Week 13: LLM Agents + Course Summary
- Week 15 (after Thanksgiving): Project Presentations

Reminder

- Project Midterm Report Grades & Feedback Released

(Recap) Prompting

- **Prompt:** initial user input/instructions given to the model to guide text generation
- Example (sentiment analysis):

$P(\text{positive} | \text{The sentiment of the sentence "I like Jackie Chan" is :})$
 $P(\text{negative} | \text{The sentiment of the sentence "I like Jackie Chan" is :})$

prompt

- Example (question answering):

$P(w | \text{Q: Who wrote the book "The Origin of Species"? A:})$

prompt

- **Prompting:** directly use trained LMs to generate text given user prompts (no fine-tuning)

For good prompting performance, we need **instruction-tuning** (later lectures)

(Recap) Prompt Engineering

- Some LMs (especially small ones) can be sensitive to specific formats of prompts
- Multiple prompts can make sense for the same task, but the resulting model performance might differ

$P_1(a) =$ It was _____. a $P_2(a) =$ Just ____! || a

$P_3(a) =$ a . All in all, it was ____.

$P_4(a) =$ a || In summary, the restaurant is ____.

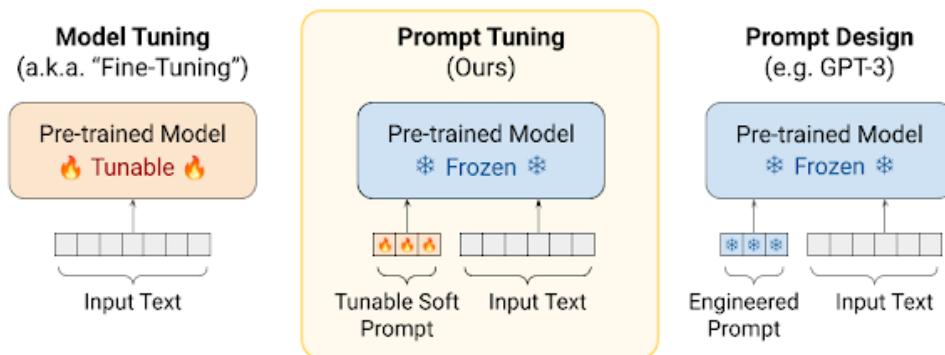
Model predicts the masked word

Prompt templates for BERT sentiment classification

- **Prompt engineering:** designing and refining prompts to achieve desired outcomes from LMs (e.g., manually tune on a validation set)
- A guide on prompt engineering: <https://www.promptingguide.ai/>

(Recap) Prompt Tuning

- **Prompt tuning:** instead of manually testing the prompt design, consider prompt tokens as learnable model parameters (“soft prompts”)
- Optimize a small amount of prompt token embeddings while keeping the LM frozen



- Prompt tuning is a parameter efficient fine-tuning (PEFT) method

(Recap) Parameter Efficient Fine-tuning (PEFT)

- Fine-tuning all model parameters is expensive

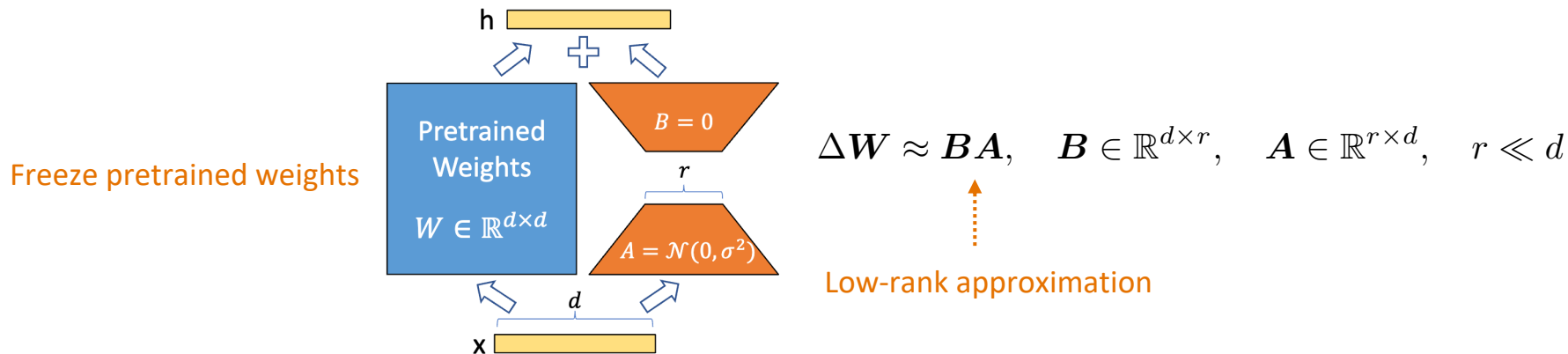
Pretrained weight
(can represent any module) $\mathbf{W}_0 \in \mathbb{R}^{d \times d}$

Fine-tuned weight $\mathbf{W}^* = \mathbf{W}_0 + \Delta\mathbf{W}, \quad \Delta\mathbf{W} \in \mathbb{R}^{d \times d}$

- Can we update only a small number of model parameters on fine-tuning data?

(Recap) Parameter Efficient Fine-tuning: LoRA

- Assume the parameter update is **low-rank**
 - Overparameterization**: large language models typically have many more parameters than strictly necessary to fit the training data
 - Empirical observation**: parameter updates in neural networks tend to be low-rank in practice
- Solution: approximate weight updates with low-rank factorization



(Recap) Large Language Models (LLMs)

- The field of LLMs is rapidly evolving!
 - In 2018, BERT-large with 340 million parameters was considered large
 - In 2019, GPT-2 with 1.5 billion parameters was considered very large
 - In 2020, GPT-3 with 175 billion parameters set a new standard for “large”
- In 2025, how should we define LLMs?
- General definition:
 - Transformer-decoder architecture (or variants) that can generate text
 - Pretrained on vast and diverse general-domain corpora
 - With (at least) billions of parameters
 - General-purpose solvers for a wide range of NLP tasks and beyond

(Recap) Decoding with LLMs

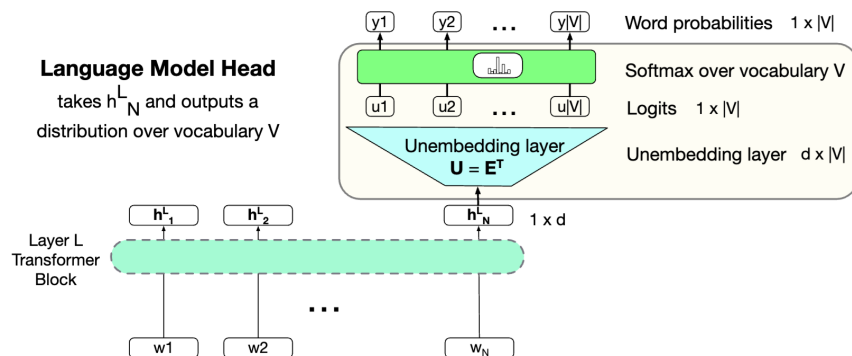
- **Decoding:** convert Transformer representations into natural language tokens
- Autoregressive decoding typically involves iterative **sampling** from LMs' output distributions, until an [EOS] token is generated

$$p_{\theta}(w|x_1, x_2, \dots, x_{i-1}) = \text{softmax}(\mathbf{U}\mathbf{h}_{i-1}) = \left[\frac{\exp(\mathbf{u}_1 \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1})}, \dots, \frac{\exp(\mathbf{u}_{|\mathcal{V}|} \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1})} \right]$$

Model parameters

Unembedding matrix

Hidden states at token $i - 1$



(Recap) Greedy Decoding

- Always pick the token with the highest probability estimated by the LM for every step

$$x_i \leftarrow \arg \max_w p_{\theta}(w | x_1, x_2, \dots, x_{i-1})$$

- Pros:
 - Simplicity: easy to implement and understand
 - Deterministic: guarantee the same output given the same input
 - Efficient: makes only one (simple) decision at each step w/o additional operations
- Cons:
 - Suboptimal solutions: may not find the globally optimal sequence
 - Lack of diversity: cannot produce multiple outputs given the same input



(Recap) Top- k Sampling

- Motivation: Instead of choosing the single most probable word to generate, sample from the top- k most likely tokens (candidates) – avoid generating low probability tokens
- k is a hyperparameter (typically 5-10)

Compute the probability distribution only over the top- k tokens

$$p_{\theta}(w|x_1, x_2, \dots, x_{i-1}) = \text{softmax}(\mathbf{U}_{\text{top-}k} \mathbf{h}_{i-1}) = \left[\frac{\exp(\mathbf{u}_1 \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^k \exp(\mathbf{u}_{\text{top-}j} \cdot \mathbf{h}_{i-1})}, \dots, \frac{\exp(\mathbf{u}_{\text{top-}k} \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^k \exp(\mathbf{u}_{\text{top-}j} \cdot \mathbf{h}_{i-1})} \right]$$

Sample from the top- k tokens $x_i \sim p_{\theta}(w|x_1, x_2, \dots, x_{i-1})$

- With $k = 1$, top- k sampling is equivalent to greedy decoding



(Recap) Nucleus (Top- p) sampling

- Top- k sampling does not account for the shape of the probability distribution
 - For the next-token distribution of “the 46th US president Joe”, top- k sampling may consider more tokens than necessary
 - For the next-token distribution of “the spacecraft”, top- k sampling may consider fewer tokens than necessary
- Nucleus sampling sets cutoff based on the top- p percent of the probability mass
- p is a hyperparameter (typically 0.9)
- Top- p vocabulary is the smallest set of words such that

$$\sum_{w \in \mathcal{V}_{\text{top-}p}} p(w|x_1, x_2, \dots, x_{i-1}) \geq p$$

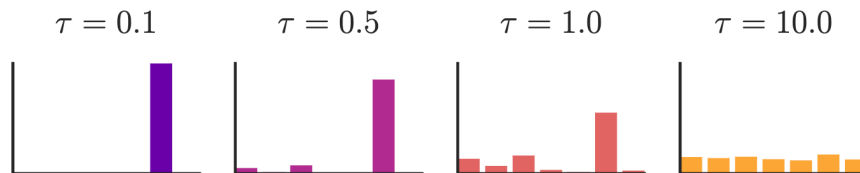
- Sample from the top- p vocabulary in a similar way as top- k sampling

(Recap) Temperature Sampling

- Intuition comes from thermodynamics
 - A system at a high temperature is flexible and can explore many possible states
 - A system at a lower temperature is likely to explore a subset of lower energy (better) states
- Reshape the probability distribution by incorporating a temperature hyperparameter

$$p_{\theta}(w|x_1, x_2, \dots, x_{i-1}) = \text{softmax}(\mathbf{U}\mathbf{h}_{i-1}/\tau) = \left[\frac{\exp(\mathbf{u}_1 \cdot \mathbf{h}_{i-1}/\tau)}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1}/\tau)}, \dots, \frac{\exp(\mathbf{u}_{|\mathcal{V}|} \cdot \mathbf{h}_{i-1}/\tau)}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1}/\tau)} \right]$$

- With $\tau \rightarrow 0$, temperature sampling approaches greedy decoding



(Recap) Practical Considerations of Decoding Algorithms

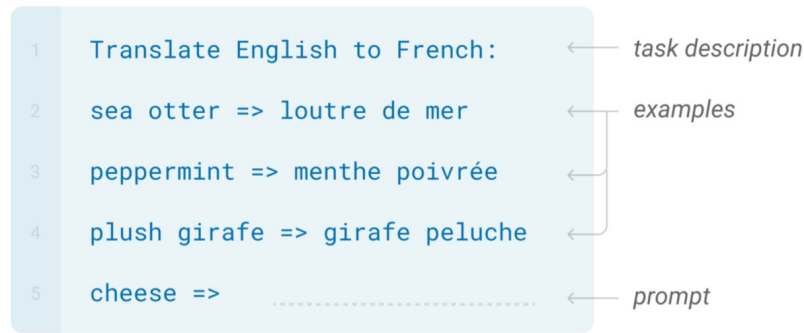
- If aiming for simplicity and efficiency without diversity requirements, use greedy decoding
- If multiple responses are required for the same input, use sampling-based decoding
 - Top- p is usually better than Top- k
 - Temperature sampling is commonly used
 - Top- p can be used together with temperature sampling

(Recap) In-context Learning

- In-context learning is a type of few-shot learning
 - User provides a few examples of input-output pairs in the prompt
 - The model uses given examples to predict the output for new, similar inputs
- First studied in the GPT-3 paper
- No model parameter updates

Few-shot

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





In-context Learning Demo

Prompt: Swap the second and the penultimate letter of the following word: pothyn

Swap the second and the penultimate letter of the following word: pothyn

∞ llama-3.3-70b-instruct

The word is "pothyn".

The second letter is "o" and the penultimate letter is "y" (the last letter is "n").

Swapping them results in "pythno" and then "pythyn".



**Wrong generation only
given the prompt**

In-context Learning Demo

Prompt: Directly answer the last one (swapping the second and the penultimate letter of the following words): tarehd -> thread, revir -> river, pothyn ->

Directly answer the last one (swapping the second and the penultimate letter of the following words): tarehd → thread, revir → river, pothyn →

∞ llama-3.3-70b-instruct

pothyn → python



**Correctly learn from the
in-context examples**



In-context Learning Demo

Prompt: How many 'r' letters are there in the following word: "strawberry"

How many 'r' letters are there in the following word: "strawberry"

llama-3.3-70b-instruct

There are 2 'r' letters in the word "strawberry".



**Wrong generation only
given the prompt**



In-context Learning Demo

Prompt: Count how many 'r' letters are there in the following words: "red": 1, "roar": 2, "strawberry":

Count how many 'r' letters are there in the following words: "red": 1, "roar": 2, "strawberry":

llama-3.3-70b-instruct

Let's count the 'r' letters in the word "strawberry":

1. s - no 'r'
2. t - no 'r'
3. r - 1 'r'
4. a - no 'r'
5. w - no 'r'
6. b - no 'r'
7. e - no 'r'
8. r - 1 'r'
9. r - 1 'r'
10. y - no 'r'

There are 3 'r' letters in the word "strawberry".



**Correctly learn from the
in-context examples**

Further Reading on In-context Learning

- [An Explanation of In-context Learning as Implicit Bayesian Inference](#) [Xie et al., 2021]
- [Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?](#) [Min et al., 2022]
- [What Can Transformers Learn In-Context? A Case Study of Simple Function Classes](#) [Garg et al., 2022]
- [What learning algorithm is in-context learning? Investigations with linear models](#) [Akyurek et al., 2023]

Agenda

- LLM Scaling Laws
- Question Answering (QA)
- Parametric Knowledge in LMs
- Hallucination

Scaling Up Pretraining Data

The Pile: 22 sub-datasets (> 800GB), a common choice for pretraining corpus

Composition of the Pile by Category

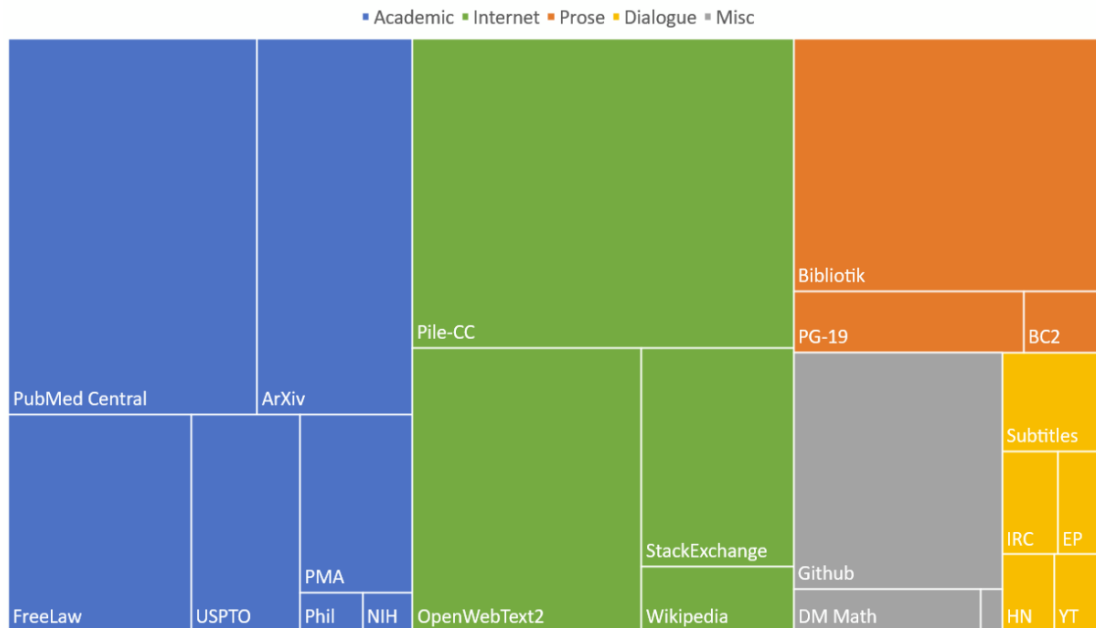
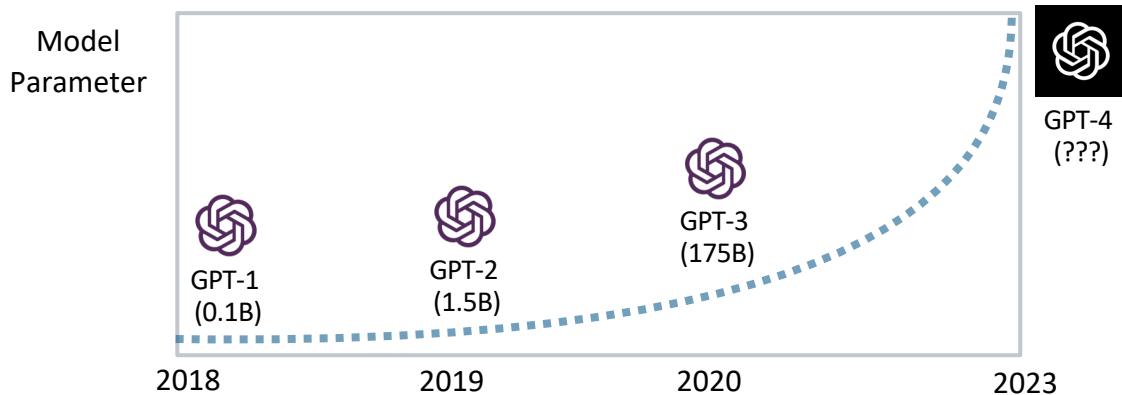


Figure source: <https://arxiv.org/pdf/2101.00027>

Scaling Up Model Sizes

- GPT-1 (2018): 12 layers, 117M parameters, trained in ~1 week
- GPT-2 (2019): 48 layers, 1.5B parameters, trained in ~1 month
- GPT-3 (2020): 96 layers, 175B parameters, trained in several months



Papers: (GPT-1) https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

(GPT-2) https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

(GPT-3) <https://arxiv.org/pdf/2005.14165.pdf>

Emergent Ability

- Larger models develop **emergent abilities**
 - Skills or capabilities that were not explicitly learned but arise as a result of model capacity
 - Larger models demonstrate surprising abilities in challenging tasks even when they were not explicitly trained for them
- Emergent capabilities typically become noticeable only when the model size reaches a certain threshold (cannot be predicted by small model's performance)

Emergent Abilities of Large Language Models

Jason Wei¹

Yi Tay¹

Rishi Bommasani²

Colin Raffel³

Barret Zoph¹

Sebastian Borgeaud⁴

Dani Yogatama⁴

Maarten Bosma¹

Denny Zhou¹

Donald Metzler¹

Ed H. Chi¹

Tatsunori Hashimoto²

Oriol Vinyals⁴

Percy Liang²

Jeff Dean¹

William Fedus¹

jasonwei@google.com

ytay@google.com

nlprishi@stanford.edu

craftel@gmail.com

barretzoph@google.com

sborgeaud@deepmind.com

dyogatama@deepmind.com

bosma@google.com

dennyzhou@google.com

metzler@google.com

edchi@google.com

thashim@stanford.edu

vinyals@deepmind.com

pliang@stanford.edu

jeff@google.com

liamfedus@google.com

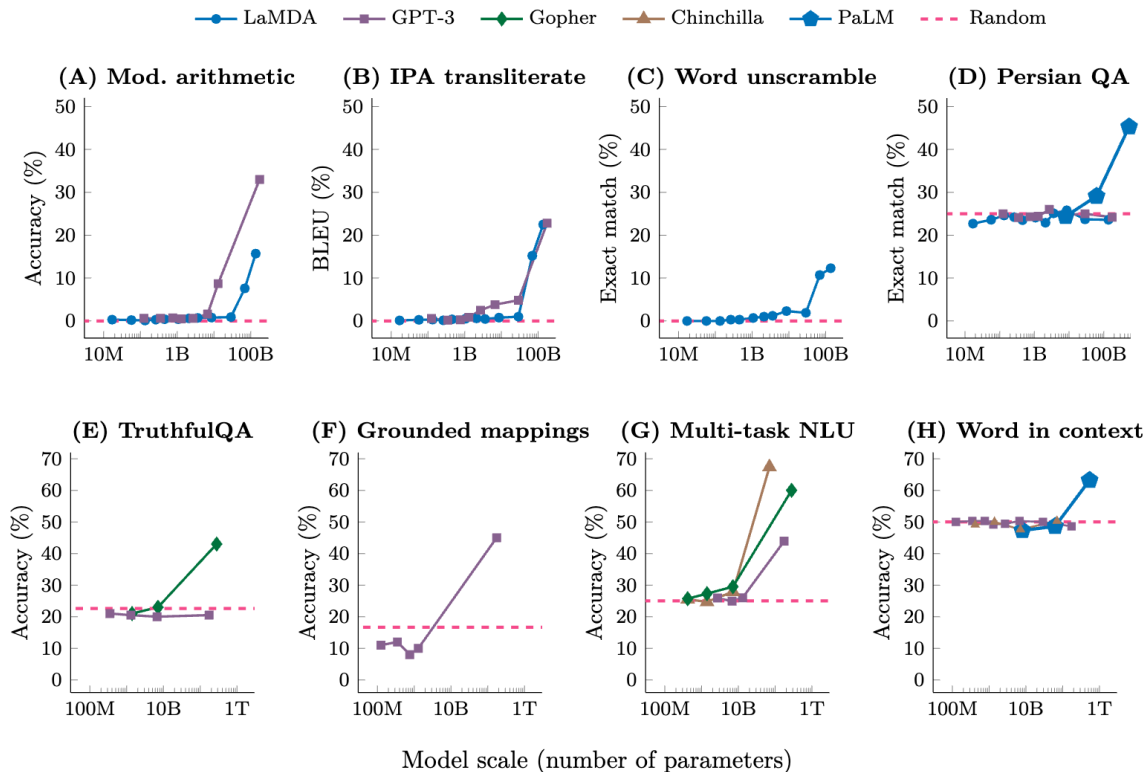
Paper: <https://arxiv.org/pdf/2206.07682>

¹ Google Research ² Stanford University ³ UNC Chapel Hill ⁴ DeepMind

Experiment Setting

- Consider the **few-shot in-context learning** paradigm
- Consider an ability to be **emergent** when a model has **random** performance until a certain scale, after which performance increases to **well-above random**
- Abilities to test
 - Arithmetic: addition, subtraction, multiplication
 - Transliteration
 - Recover a word from its scrambled letters
 - Persian question answering
 - Question answering (truthfully)
 - Grounded conceptual mappings
 - Multi-task understanding (math, history, law, ...)
 - Contextualized semantic understanding

Performance vs. Model Scale



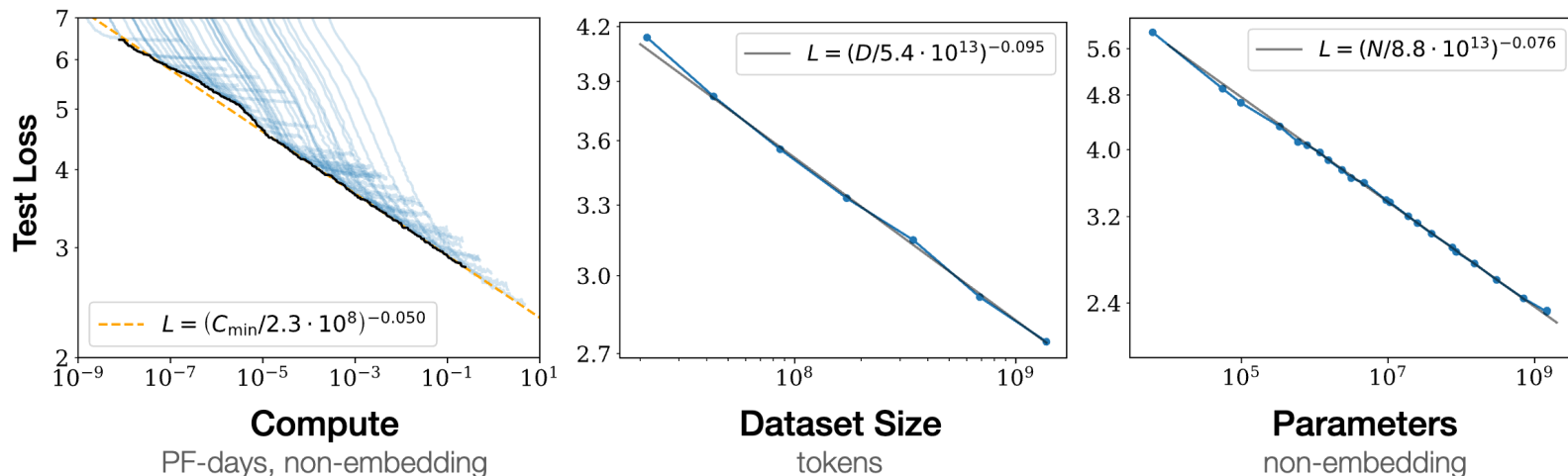
Models exhibit random performance until a certain scale, after which performance significantly increases

Scaling Laws of LLMs

- (Pretrained) LLM performance is mainly determined by 3 factors
 - Model size: the number of parameters
 - Dataset size: the amount of training data
 - Compute: the amount of floating point operations (FLOPs) used for training
- Scaling up LLMs involves scaling up the 3 factors
 - Add more parameters (adding more layers or having more model dimensions or both)
 - Add more data
 - Train for more iterations
- **Scaling laws:** study the correlation between the cross-entropy language modeling loss and the above three factors
- How to optimally allocate a fixed compute budget?

Scaling Laws of LLMs

Performance has a power-law relationship with each of the three scale factors (model size, dataset size, compute) when not bottlenecked by the other two

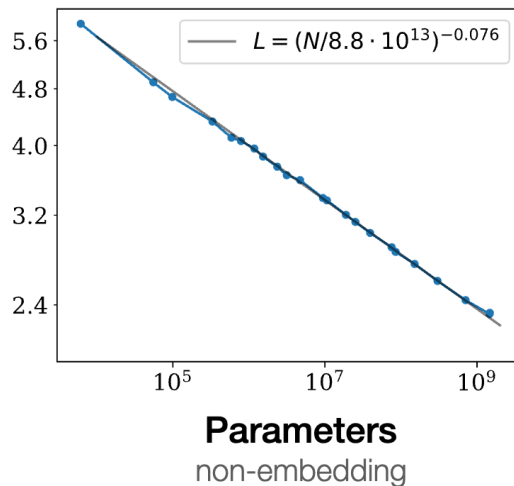


Scaling Model Parameters

- Language model loss vs. models with a limited number of parameters (N)
 - Only count non-embedding parameters
 - Infinite compute: trained to convergence
 - Infinite dataset: trained with sufficiently large datasets
- Performance depends strongly on scale, weakly on model shape (depth vs. width)

$$\mathcal{L}(N) = \left(\frac{N_c}{N} \right)^{\alpha_N}, \quad \alpha_N \approx 0.076, \quad N_c \approx 8.8 \times 10^{13}$$


 Model parameters
 (non-embedding)

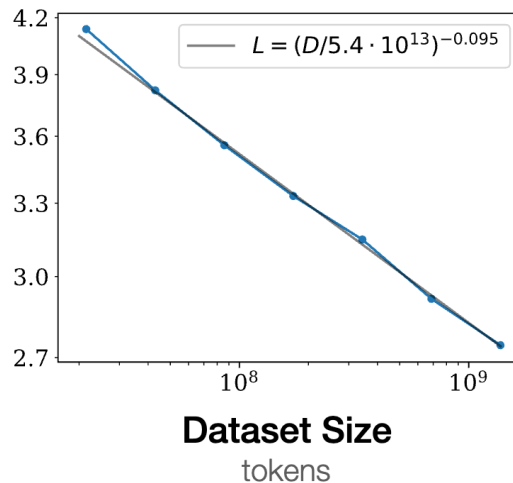


Scaling Dataset Size

- Language model loss vs. a limited dataset size (D)
 - Infinite model size: sufficiently large model
 - With appropriate early stopping: avoid overfitting to the training data

$$\mathcal{L}(D) = \left(\frac{D_c}{D} \right)^{\alpha_D}, \quad \alpha_D \approx 0.095, \quad D_c \approx 5.4 \times 10^{13}$$

↑
Dataset size
(# of tokens)

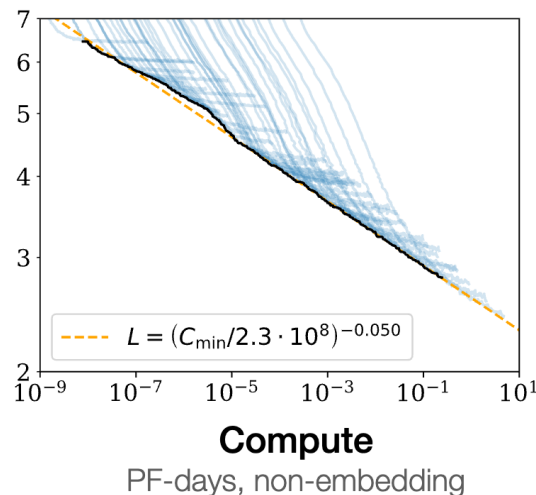


Scaling Training Compute

- Language model loss vs. a limited amount of compute (C)
 - Infinite dataset size: sufficiently large training corpus
 - Optimal model size: can effectively learn the data and not excessively compute-consuming

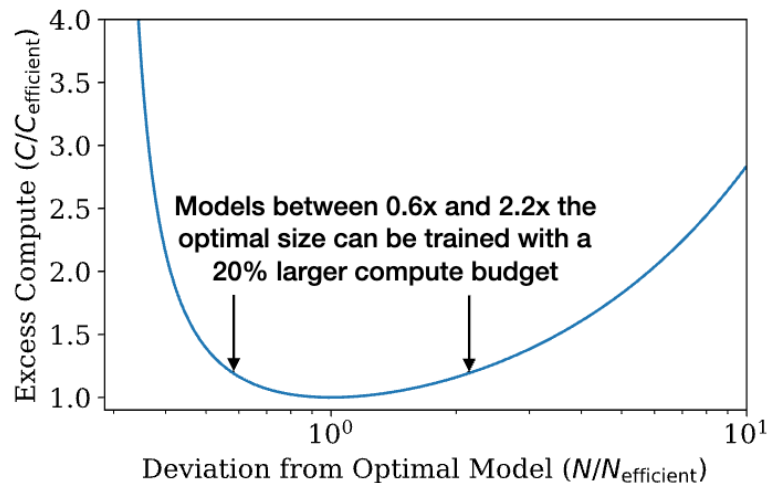
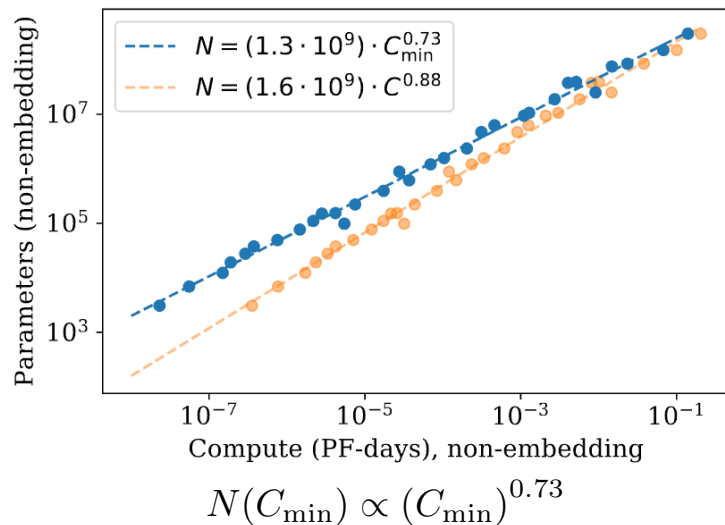
$$\mathcal{L}(C) = \left(\frac{C_c}{C} \right)^{\alpha_C}, \quad \alpha_C \approx 0.050, \quad C_c \approx 3.1 \times 10^8$$


 Compute
 (# Peta-FLOP days)



Optimal Model Size

- Given a specific amount of training compute C , what's the optimal model size $N(C)$ that leads to minimal language modeling loss?
- $N(C)$ can be fit with a power-law wrt C
- Additional compute needs to be used when model size is suboptimal





Further Reading on Scaling LLMs

- [Training Compute-Optimal Large Language Models](#) [Hoffmann et al., 2022]
- [Scaling Data-Constrained Language Models](#) [Muennighoff et al., 2023]
- [Are Emergent Abilities of Large Language Models a Mirage?](#) [Schaeffer et al., 2023]

Agenda

- LLM Scaling Laws
- Question Answering (QA)
- Parametric Knowledge in LMs
- Hallucination

Introduction to Question Answering

- **Question Answering (QA):** build systems that can automatically answer questions posed by humans in natural language
- Categorization by application domain: closed-domain vs. open-domain QA
- **Closed-domain QA:** answer questions within a specific domain
 - Example: medical, legal, technical fields
 - Models are trained on specialized knowledge to be highly accurate within their domain
- **Open-domain QA:** answer questions from any domain
 - Typically rely on vast (external) knowledge sources like the web or large text corpora
 - Most LLM applications consider open-domain QA settings

Introduction to Question Answering

- **Question Answering (QA):** build systems that can automatically answer questions posed by humans in natural language
- Categorization by modeling approach: extractive vs. abstractive QA
- **Extractive QA:** output a span of text extracted directly from a given context
 - A natural language understanding task (reading comprehension)
 - Example: context: “The human brain contains approximately 86 billion neurons” Q: “How many neurons are in the human brain?” A: “86 billion”
 - Can be done with encoder-only LMs (e.g., BERT)
- **Abstractive QA:** synthesize the answer in its own words (rephrasing/summarizing)
 - Example: context: “Albert Einstein published his theory of special relativity which introduced the famous equation $E=mc^2$, which relates energy (E) to mass (m) and the speed of light (c)” Q: “What did Einstein contribute to physics?” A: “Einstein made significant contributions to the theory of special relativity which established the relationship between energy and mass”
 - Need to use a generative LM (e.g., GPT)

Introduction to Question Answering

- **Question Answering (QA):** build systems that can automatically answer questions posed by humans in natural language
- Categorization by access to external source: closed-book vs. open-book QA
- **Closed-book QA:** answer questions without access to any external information
 - Accuracy depends heavily on how well the training data covered the relevant information
 - Similar to a human answering a question from memory without looking anything up
- **Open-book QA:** can access external knowledge source to answer the questions
 - Typically using retrieval from reliable external sources that contain
 - Similar to a human answering a question by looking it up in a book or online resource

Agenda

- LLM Scaling Laws
- Question Answering (QA)
- Parametric Knowledge in LMs
- Hallucination

Prompting LMs: Parametric Knowledge

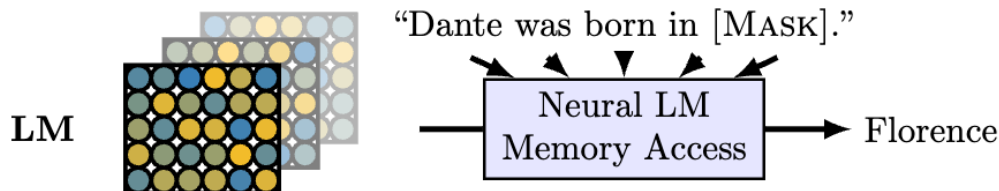
- LMs have learned from a lot of facts in their pretraining data
- LMs can be directly prompted to generate answers to factoid questions (Closed-book QA setting)
- Example:

$P(w|$ Q: Who wrote the book "The Origin of Species"? A:) prompt

- Since prompting LLMs only relies on the information stored within the parameters of the model itself, this kind of knowledge is called **parametric knowledge**

Language Model as Knowledge Bases

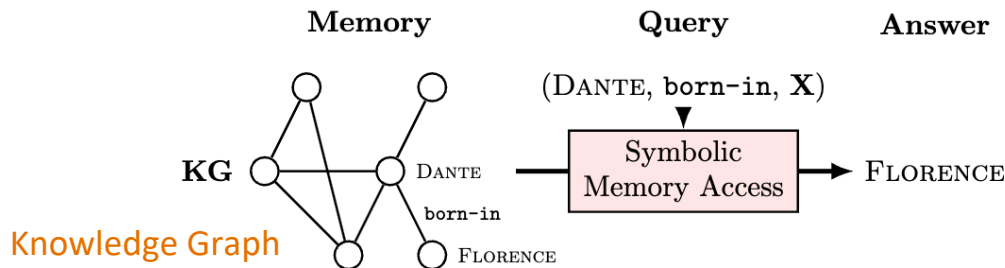
- **Acquisition:** LM's knowledge is derived from the vast amount of pretraining data
- **Access:** information is accessed through natural language prompts
- **Update/maintenance:** re-training/fine-tuning the model with new data
- **Pros:**
 - Handle a wide range of natural language queries with contextual understanding
 - Generalize to unseen queries not seen during training
- **Cons:**
 - May produce incorrect/outdated information
 - Lack interpretability/transparency





(Real) Knowledge Bases

- **Acquisition:** manually constructed by human annotators
- **Access:** information is accessed through queries in specific formats
- **Update/maintenance:** adding/modifying/deleting entries (incrementally) by humans
- **Pros:**
 - Precise & verifiable
- **Cons:**
 - Not able to handle natural language
 - Require massive human efforts to construct & maintain

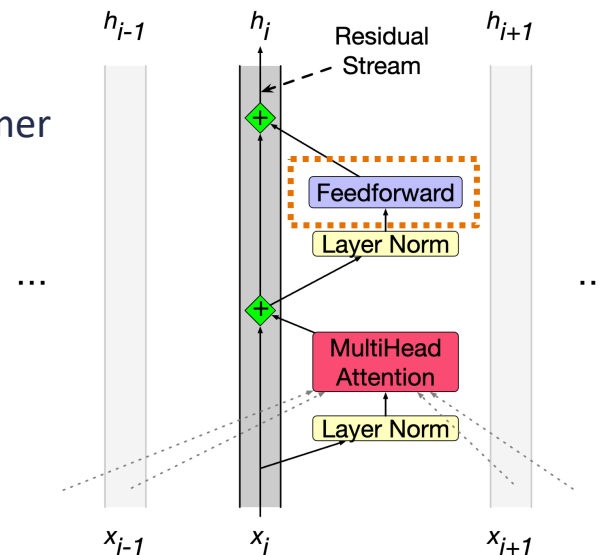


Feedforward Parameters in Transformer

- FFN in Transformer is a 2-layer network (one hidden layer, two weight matrices)

$$\text{FFN}(\mathbf{x}_i) = \text{ReLU}(\mathbf{x}_i \mathbf{W}_1) \mathbf{W}_2$$

- FFN constitutes $\sim 2/3$ of the total parameters of Transformer



Feedforward Parameters Are Neural Memories

Viewing FFN as key-value memories

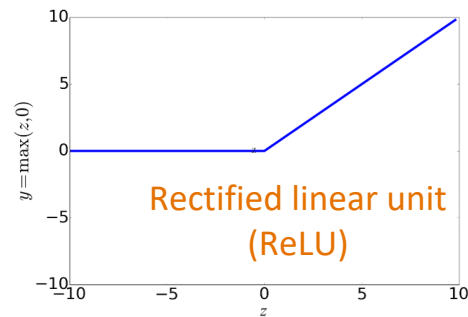
$$\text{FFN}(\mathbf{x}_i) = \text{ReLU}(\mathbf{x}_i \mathbf{W}_1) \mathbf{W}_2$$

↓

$\mathbf{x}_i \in \mathbb{R}^{d_1}$

$$\text{FFN}(\mathbf{x}_i) = \text{ReLU}(\mathbf{x}_i \mathbf{K}) \mathbf{V}$$

$\mathbf{K} \in \mathbb{R}^{d_1 \times d_2} \quad \mathbf{V} \in \mathbb{R}^{d_2 \times d_1}$



key vectors (column vectors in \mathbf{K}) act as **pattern detectors** over the input sequence

value vectors (row vectors in \mathbf{V}) represent **distributions over the output vocabulary**

$$\text{FFN}(\mathbf{x}_i) = \sum_{j=1}^{d_2} \text{ReLU}(\mathbf{x}_i \cdot \mathbf{k}_j) \mathbf{v}_j$$

weights of value vectors

Memory Keys Correspond to Input Patterns

Each individual key vector corresponds to a specific pattern over the input prefix

Key	Pattern	Example trigger prefixes
k_{449}^1	Ends with “substitutes” (shallow)	<i>At the meeting, Elton said that “for artistic reasons there could be no substitutes In German service, they were used as substitutes Two weeks later, he came off the substitutes</i>
k_{2546}^6	Military, ends with “base”/“bases” (shallow + semantic)	<i>On 1 April the SRSG authorised the SADF to leave their bases Aircraft from all four carriers attacked the Australian base Bombers flying missions to Rabaul and other Japanese bases</i>
k_{2997}^{10}	a “part of” relation (semantic)	<i>In June 2012 she was named as one of the team that competed He was also a part of the Indian delegation Toy Story is also among the top ten in the BFI list of the 50 films you should</i>
k_{2989}^{13}	Ends with a time range (semantic)	<i>Worldwide, most tornadoes occur in the late afternoon, between 3 pm and 7 Weekend tolls are in effect from 7:00 pm Friday until The building is open to the public seven days a week, from 11:00 am to</i>
k_{1935}^{16}	TV shows (semantic)	<i>Time shifting viewing added 57 percent to the episode’s The first season set that the episode was included in was as part of the From the original NBC daytime version , archived</i>



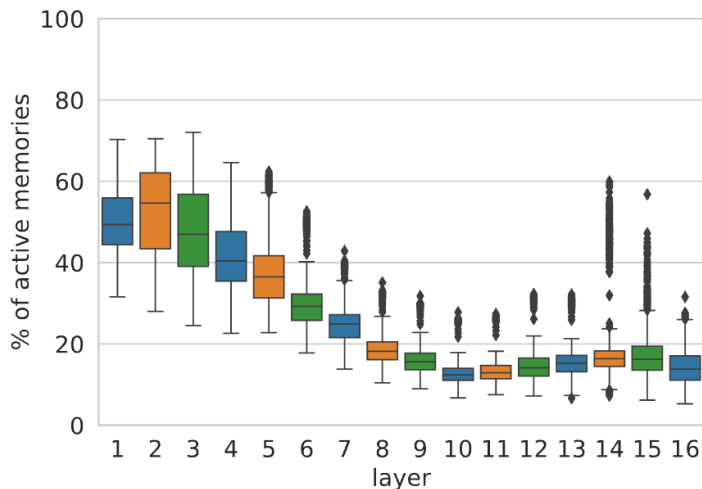
Memory Values Correspond to Output Tokens

Each value vector (roughly) matches a predicted token distribution

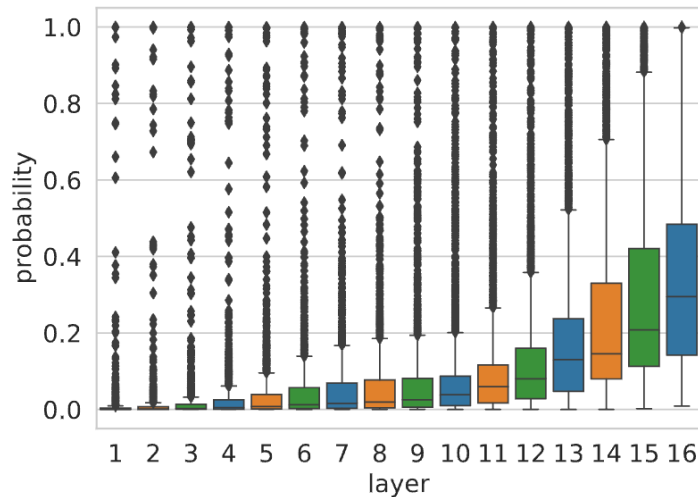
Value	Prediction	Trigger example
\mathbf{v}_{222}^{15}	<i>each</i>	<i>But when bees and wasps resemble each</i>
\mathbf{v}_{752}^{16}	<i>played</i>	<i>Her first role was in Vijay Lalwani's psychological thriller Karthik Calling Karthik, where Padukone was cast as the supportive girlfriend of a depressed man (played)</i>
\mathbf{v}_{2601}^{13}	<i>extratropical</i>	<i>Most of the winter precipitation is the result of synoptic scale, low pressure weather systems (large scale storms such as extratropical)</i>
\mathbf{v}_{881}^{15}	<i>part</i>	<i>Comet served only briefly with the fleet, owing in large part</i>
\mathbf{v}_{2070}^{16}	<i>line</i>	<i>Sailing from Lorient in October 1805 with one ship of the line</i>
\mathbf{v}_{3186}^{12}	<i>jail</i>	<i>On May 11, 2011, four days after scoring 6 touchdowns for the Slaughter, Grady was sentenced to twenty days in jail</i>

Memory Aggregation

- “Active” memories (memory vectors with non-zero coefficients) are typically sparse
- The residual connection sequentially refines token prediction from layer to layer



Fraction of active memory
units across layers



Output token probability is
gradually refined across layers

Further Reading on LLM Parametric Knowledge

- [How Much Knowledge Can You Pack Into the Parameters of a Language Model?](#) [Roberts et al., 2020]
- [Extracting Training Data from Large Language Models](#) [Carlini et al., 2021]
- [Locating and Editing Factual Associations in GPT](#) [Meng et al., 2022]

Agenda

- LLM Scaling Laws
- Question Answering (QA)
- Parametric Knowledge in LMs
- Hallucination

Hallucination

- **Hallucination:** LM generates information that is factually incorrect, misleading, or fabricated, even though it may sound plausible or convincing
- Why does hallucination happen?
 - Limited knowledge: LLMs are trained on finite datasets, which don't have access to all possible information; when asked about topics outside their training data, they may generate plausible-sounding but incorrect responses
 - Overgeneralization: LLMs may apply patterns they've learned from one context to another where they don't apply, leading to incorrect conclusions
 - Lack of common sense: While LLMs can process and generate human-like text, they often lack the ability to apply commonsense reasoning to their outputs
 - ...

Hallucination Examples

- **(Limited knowledge)** Q: “What were the main features of the iPhone 15 Pro Max?”
LLM (trained before 2023): “The iPhone 15 Pro Max features a revolutionary holographic display, quantum computing chip, and telepathic user interface.”
- **(Overgeneralization)** Q: “How do you form the past tense in Japanese?”
LLM: “In Japanese, you typically add '-ed' to the end of verbs to form the past tense, just like in English.” (incorrect)
- **(Lack of common sense)** Q: “How many tennis balls can fit in a typical smartphone?”
LLM: “Approximately 15-20 tennis balls can fit in a typical smartphone, depending on the model and screen size.”

what's your knowledge cutoff date?



My knowledge cutoff date is October 2023. This means I don't have information on events or developments that have occurred after that time. How can I assist you with your question?

Concerns About Hallucination

BAKER DONELSON

[Home](#) > [Publications](#) > [The Perils of Legal Hallucinations and the Need for AI Training for Your In-House Legal Team](#)



Generative AI is rewriting how lawyers work at record speed, but is also quietly flooding Courts with phantom citations and invented case law along the way. In recent months, a number of judges across the country have sanctioned attorneys for submitting briefs laced with fictitious case law conjured by generative AI tools like ChatGPT. These so-called "hallucinations" aren't just embarrassing, they can constitute professional misconduct, jeopardize your company's interests, and damage the credibility of the legal profession in an already skeptical courtroom environment. In fact, since mid-2023, more than 120 cases of AI-driven legal "hallucinations" have been identified, with at least 58 occurring so far in 2025.

Figure source: <https://www.bakerdonelson.com/the-perils-of-legal-hallucinations-and-the-need-for-ai-training-for-your-in-house-legal-team>

Further Reading on Hallucination

- [LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples](#) [Yao et al., 2023]
- [Towards Mitigating Hallucination in Large Language Models via Self-Reflection](#) [Ji et al., 2023]
- [Hallucination is Inevitable: An Innate Limitation of Large Language Models](#) [Xu et al., 2024]



Thank You!

Yu Meng

University of Virginia

yumeng5@virginia.edu