In-context Learning and Scaling Laws

Yu Meng
University of Virginia
yumeng5@virginia.edu

Oct 18, 2024

Reminder

- Midterm report due today! (Guideline: https://docs.google.com/document/d/12-f2KQRH2kYBohxJLj_E6gzfj1vulmnuaEVBbyXBAiY/edit?usp=sharing)
- Assignment 4 has been released

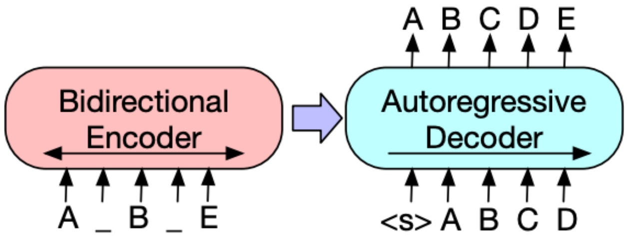
Join at
slido.com
#1940 876

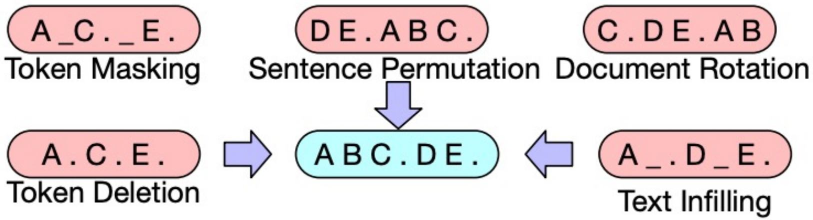
Overview of Course Contents

- 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 and Neural Language Models
- Week 6-7: Language Modeling with Transformers (Pretraining + Fine-tuning)
- Week 8: Large Language Models (LLMs) & In-context Learning
- Week 9-10: Knowledge in LLMs and Retrieval-Augmented Generation (RAG)
- Week 11: LLM Alignment
- Week 12: Language Agents
- Week 13: Recap + Future of NLP
- Week 15 (after Thanksgiving): Project Presentations

(Recap) Encoder-Decoder Architecture: BART

- Pretraining: Apply a series of noising schemes (e.g., masks, deletions, permutations…) to input sequences and train the model to recover the original sequences
- Fine-tuning:
  - For NLU tasks: Feed the same input into the encoder and decoder, and use the final decoder token for classification
  - For NLG tasks: The encoder takes the input sequence, and the decoder generates outputs autoregressively

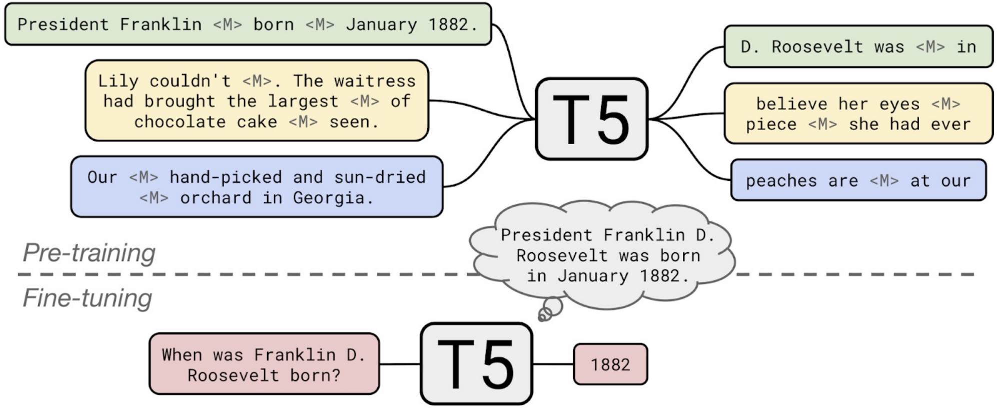




BART: https://arxiv.org/pdf/1910.13461.pdf

(Recap) Encoder-Decoder Architecture: T5

- T5: **T**ext-**to**-**T**ext **T**ransfer **T**ransformer
- Pretraining: Mask out spans of texts; generate the original spans
- Fine-tuning: Convert every task into a sequence-to-sequence generation problem
- We'll see this model again in the instruction tuning lectures



T5: https://arxiv.org/pdf/1910.10683

(Recap) Encoder-Decoder vs. Decoder-Only

- Modern LLMs are mostly based on the decoder-only Transformer architecture
- Simplicity:
  - Decoder-only models are simpler in structure (one Transformer model)
  - Encoder-decoder models require two Transformer models
- Efficiency:
  - Decoder-only models are more parameter-efficient for text generation
  - Encoder-decoder models' encoder part does not contribute to generation
- Scalability:
  - Decoder-only models scale very well with increased model size and data
  - Encoder-decoder models do not outperform decoder-only models at large model sizes

(Recap) Prompting

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

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

prompt

- Example (question answering):

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

prompt

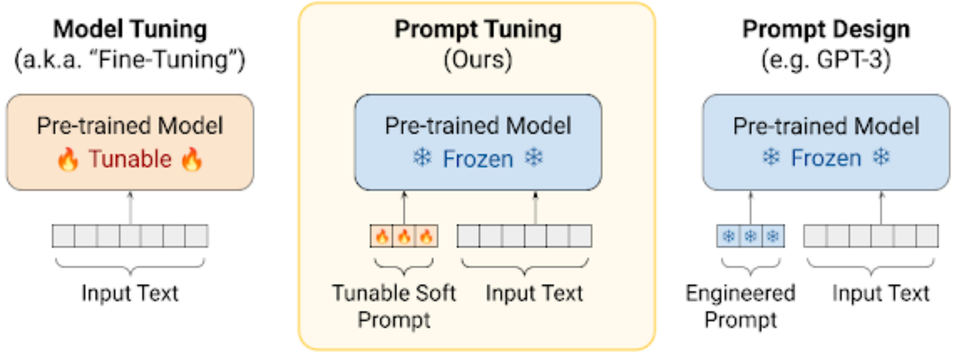
- **Prompting**: directly use trained LMs to generate text given user prompts (no fine-tuning)

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

Example source: https://web.stanford.edu/~jurafsky/slp3/10.pdf

(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

Figure source: https://www.googblogs.com/guiding-frozen-language-models-with-learned-soft-prompts/

UNIVERSITY of VIRGINIA

Join at
slido.com
#1940 876

(Recap) Parameter Efficient Fine-tuning (PEFT)

- Fine-tuning all model parameters is expensive

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

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

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

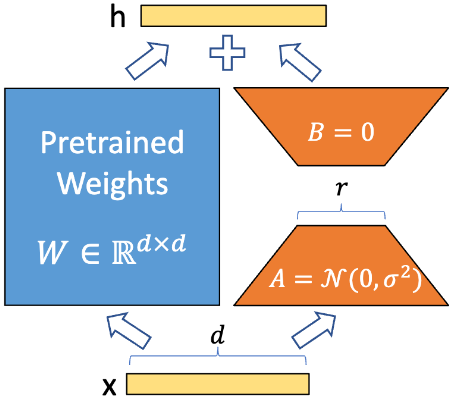
UNIVERSITY of VIRGINIA

Join at slido.com #1940 876

(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

Freeze pretrained weights



$$\Delta \boldsymbol{W} \approx \boldsymbol{B}\boldsymbol{A}, \quad \boldsymbol{B} \in \mathbb{R}^{d\times r}, \quad \boldsymbol{A} \in \mathbb{R}^{r\times d}, \quad r \ll d$$

Low-rank approximation

LoRA: https://arxiv.org/pdf/2106.09685

UNIVERSITY of VIRGINIA

Join at
slido.com
#1940 876

(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 2024, 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

Agenda

- Large Language Models (LLMs) for Text Generation
- In-context Learning
- Scaling Up LLMs

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

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

Model parameters Unembedding matrix Hidden states at token $i - 1$

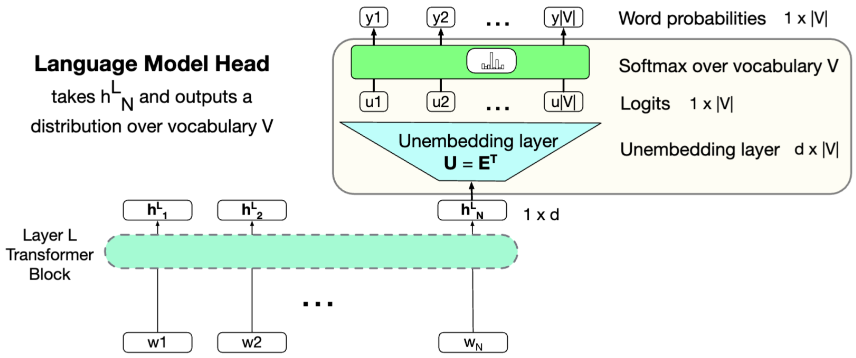


Figure source: https://web.stanford.edu/~jurafsky/slp3/9.pdf

Greedy Decoding

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

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

- 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

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

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

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

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

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

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

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

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

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

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

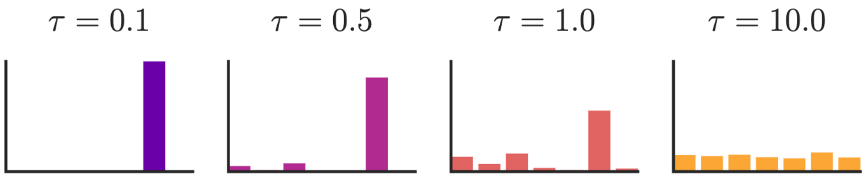


Figure source: https://arxiv.org/pdf/1611.01144v5

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

Join at
slido.com
#1940 876

Agenda

- Large Language Models (LLMs) for Text Generation
- In-context Learning
- Scaling Up LLMs

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

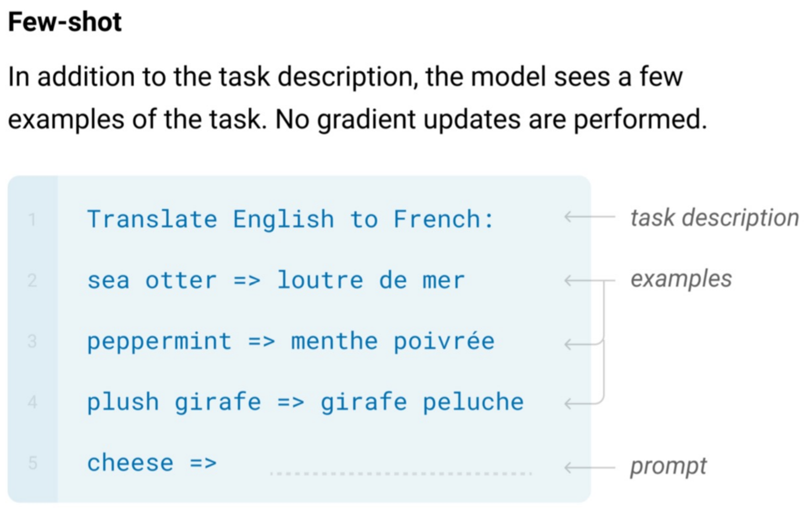
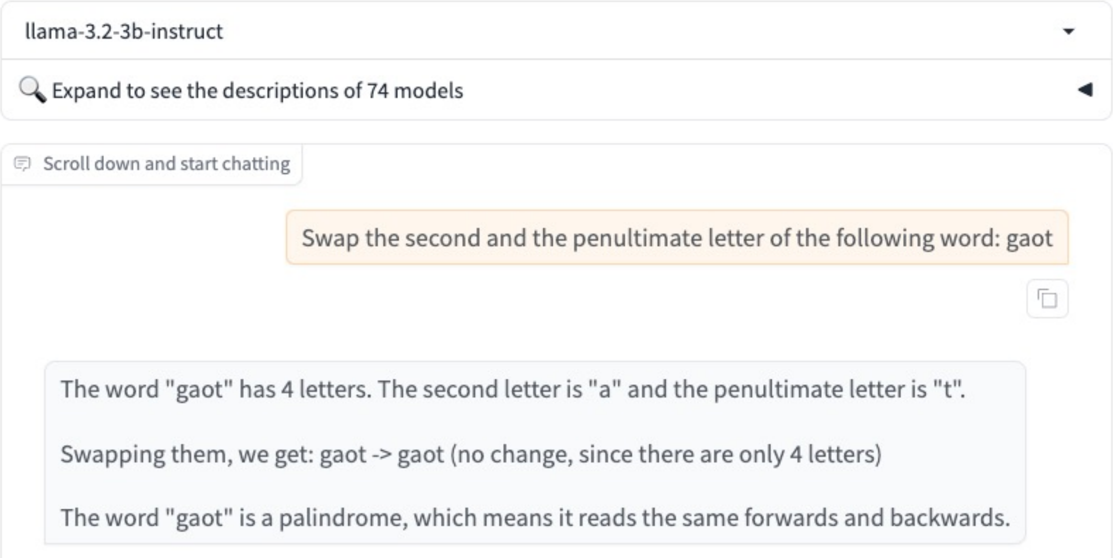


Figure source: https://arxiv.org/pdf/2005.14165

In-context Learning Demo

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



Wrong generation only given the prompt

Generated with greedy decoding
(temperature = 0)

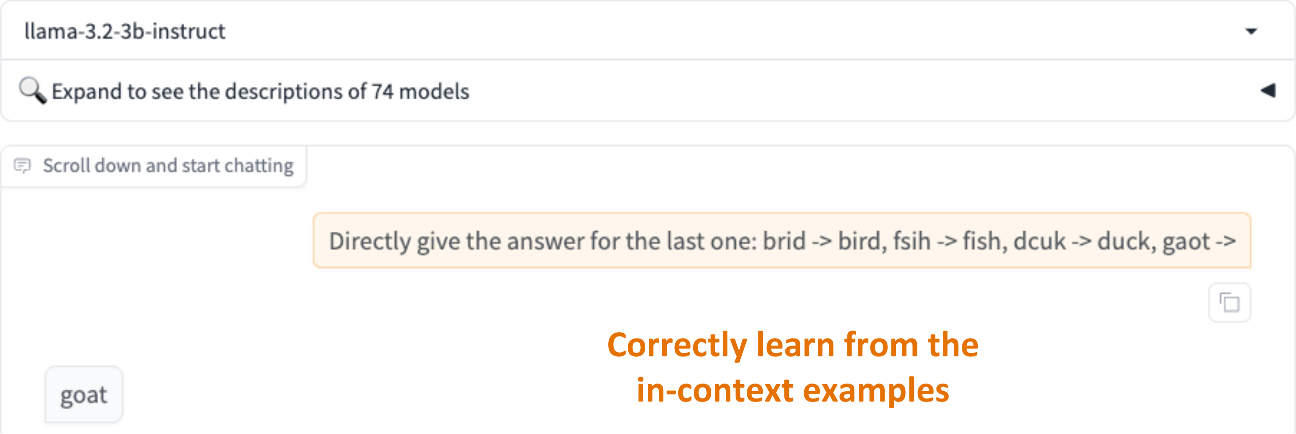
Figure source: https://lmarena.ai/?model=llama-3.2-3b-instruct

UNIVERSITY of VIRGINIA

Join at
slido.com
#1940 876

In-context Learning Demo

Prompt: Directly give the answer for the last one: brid -> bird, fsih -> fish, dcuk -> duck, gaot ->



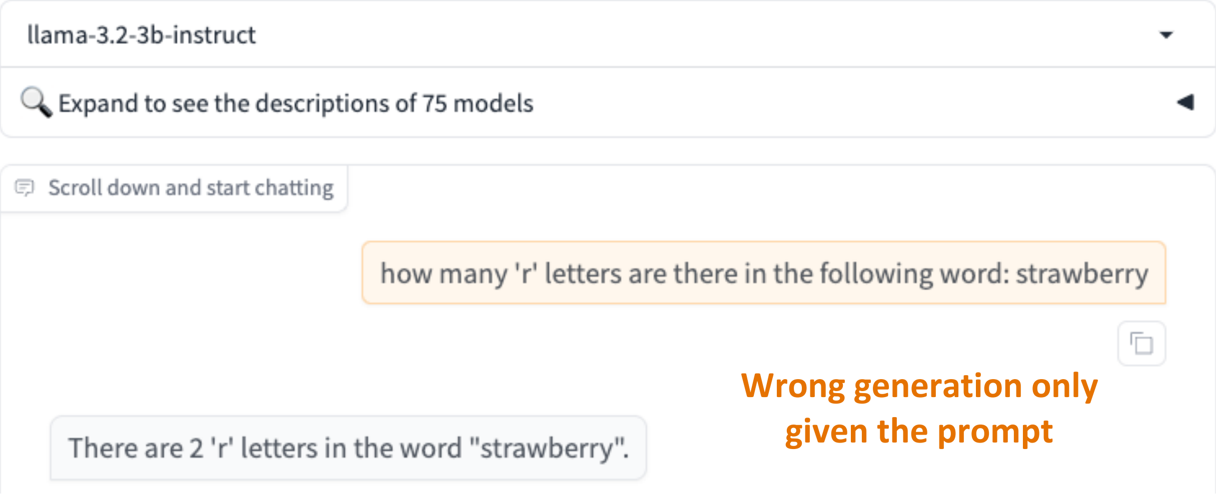
Correctly learn from the in-context examples

Generated with greedy decoding (temperature = 0)

Figure source: https://lmarena.ai/?model=llama-3.2-3b-instruct

In-context Learning Demo

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



Wrong generation only given the prompt

Generated with greedy decoding
(temperature = 0)

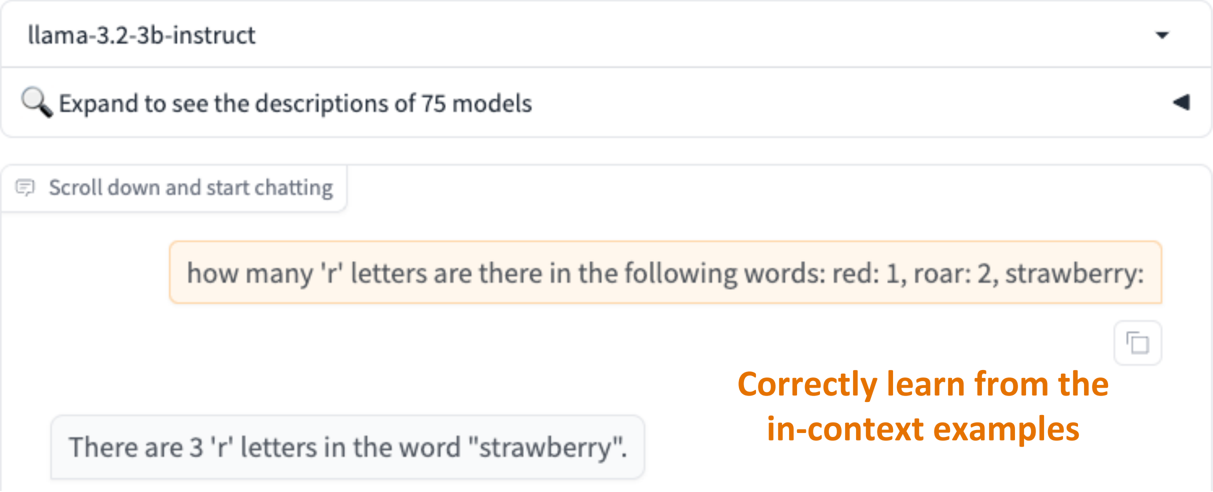
Figure source: https://lmarena.ai/?model=llama-3.2-3b-instruct

UNIVERSITY of VIRGINIA

Join at
slido.com
#1940 876

In-context Learning Demo

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



Correctly learn from the in-context examples

Generated with greedy decoding
(temperature = 0)

Figure source: https://lmarena.ai/?model=llama-3.2-3b-instruct

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

- Large Language Models (LLMs) for Text Generation
- In-context Learning
- Scaling Up LLMs

Scaling Up Pretraining Data

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

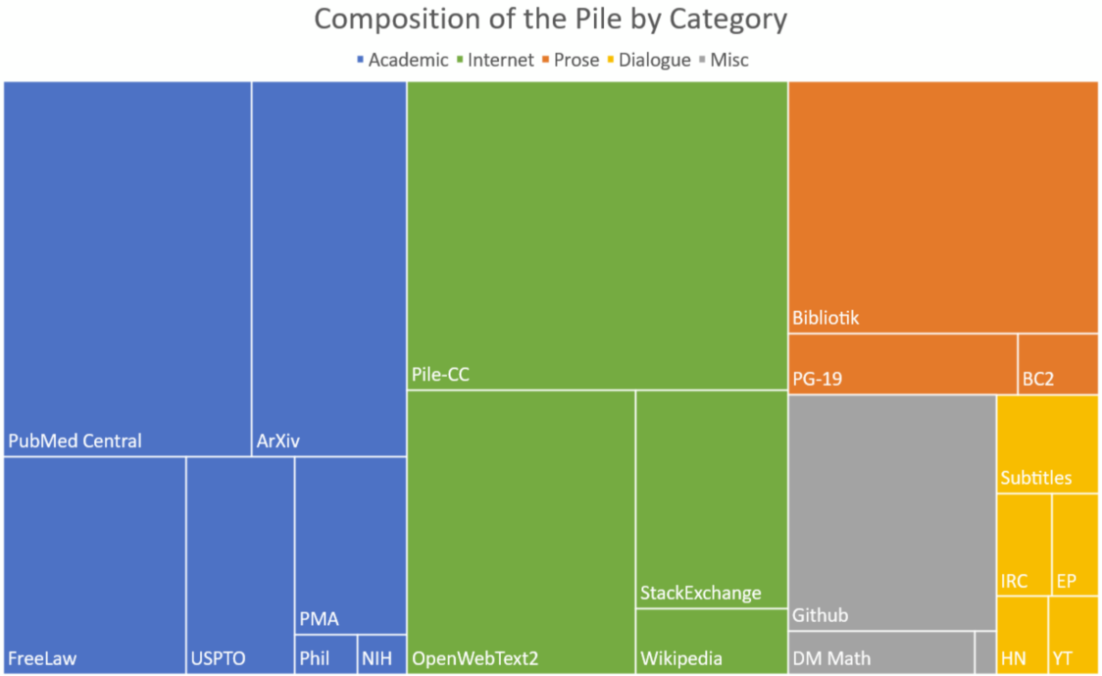


Figure source: https://arxiv.org/pdf/2101.00027

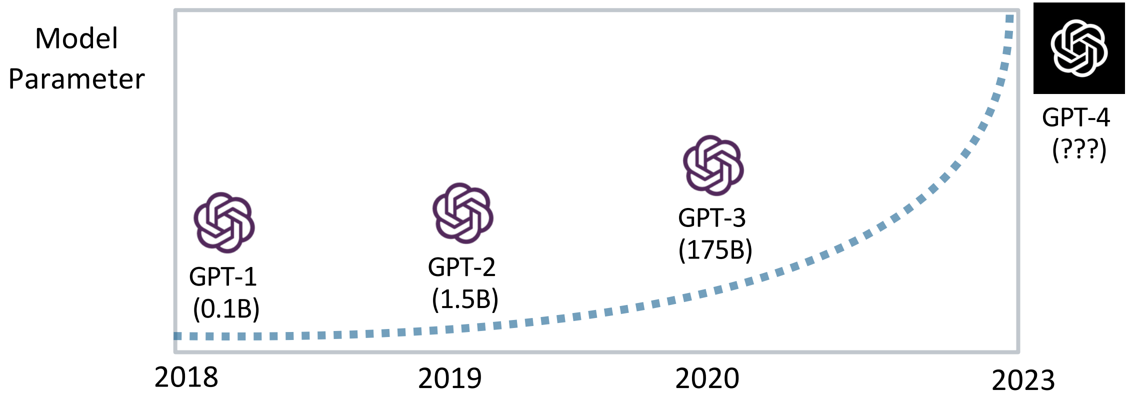
Broader Range of Knowledge by Scaling Up Data

- In my free time, I like to **{<u>run</u>, banana}** (*Grammar*)
- I went to the zoo to see giraffes, lions, and **{<u>zebras</u>, spoon}** (*Lexical semantics*)
- The capital of Denmark is **{<u>Copenhagen</u>, London}** (*World knowledge*)
- I was engaged and on the edge of my seat the whole time. The movie was **{<u>good</u>, bad}** (*Sentiment analysis*)
- The word for “pretty” in Spanish is **{<u>bonita</u>, hola}** (*Translation*)
- 3 + 8 + 4 = **{<u>15</u>, 11}** (*Math*)
- …

Examples from: https://docs.google.com/presentation/d/1hQUd3pF8_2Gr2Obc89LKjmHL0DlH-uof9M0yFVd3FA4/edit#slide=id.g28e2e9aa709_0_1

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 Wei[1] *jasonwei@google.com*
Yi Tay[1] *yitay@google.com*
Rishi Bommasani[2] *nlprishi@stanford.edu*
Colin Raffel[3] *craffel@gmail.com*
Barret Zoph[1] *barretzoph@google.com*
Sebastian Borgeaud[4] *sborgeaud@deepmind.com*
Dani Yogatama[4] *dyogatama@deepmind.com*
Maarten Bosma[1] *bosma@google.com*
Denny Zhou[1] *dennyzhou@google.com*
Donald Metzler[1] *metzler@google.com*
Ed H. Chi[1] *edchi@google.com*
Tatsunori Hashimoto[2] *thashim@stanford.edu*
Oriol Vinyals[4] *vinyals@deepmind.com*
Percy Liang[2] *pliang@stanford.edu*
Jeff Dean[1] *jeff@google.com*
William Fedus[1] *liamfedus@google.com*

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

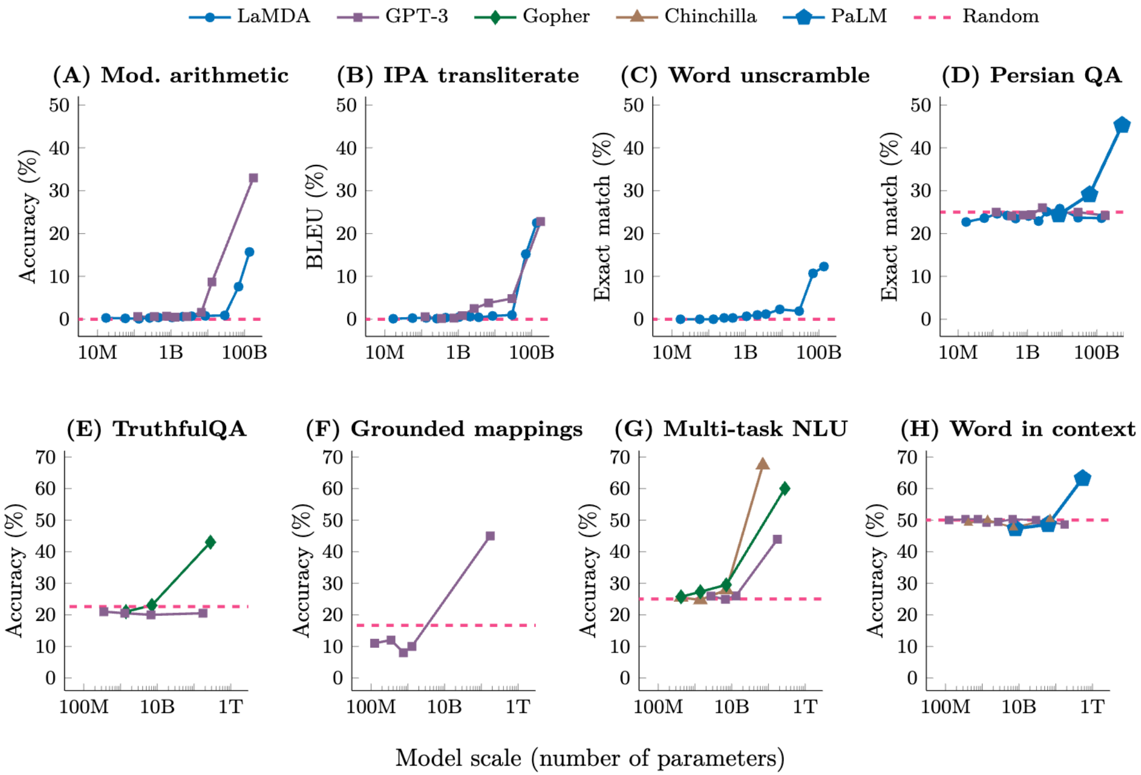
Paper: https://arxiv.org/pdf/2206.07682

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

Join at
slido.com
#1940 876



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

Figure source: https://arxiv.org/pdf/2206.07682

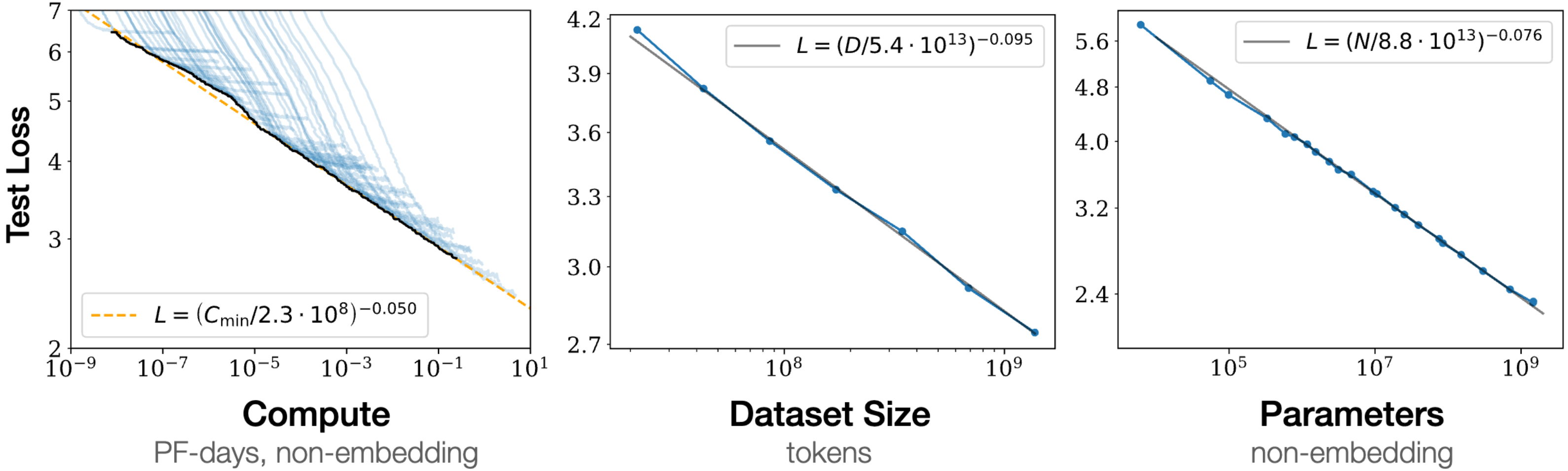
Scaling Laws of LLMs

- (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?

Join at
slido.com
#1940 876

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



Paper: https://arxiv.org/pdf/2001.08361

Summary: Large Language Models (LLMs)

- Rough definition:
  - Transformer-decoder architecture
  - Pretrained on vast and diverse general-domain corpora
  - Billions of parameters
  - General-purpose NLP task solvers
- In-context learning:
  - A unique learning paradigm in LLMs
  - No parameter updates
  - Learn from the provided few-shot demonstrations in context

Summary: LLMs Decoding

- Various decoding algorithms
  - Greedy decoding
  - Top-p (Nucleus) sampling
  - Top-k sampling
  - Temperature sampling
- Greedy decoding is most commonly used for its simplicity and efficiency
- If generation diversity is required, top-p sampling is usually used together with temperature sampling

Summary: LLMs Scaling

- LLMs exhibit emergent abilities
  - Noticeable only when the model size reaches a certain threshold
  - Cannot be extrapolated from small model performance
- Scaling up LLMs involves three factors
  - Model size
  - Dataset size
  - Compute
- Language modeling loss has a power-law relationship with each of the three scale factors

Thank You!

Yu Meng
University of Virginia
yumeng5@virginia.edu