# Foundations of Natural Language Processing

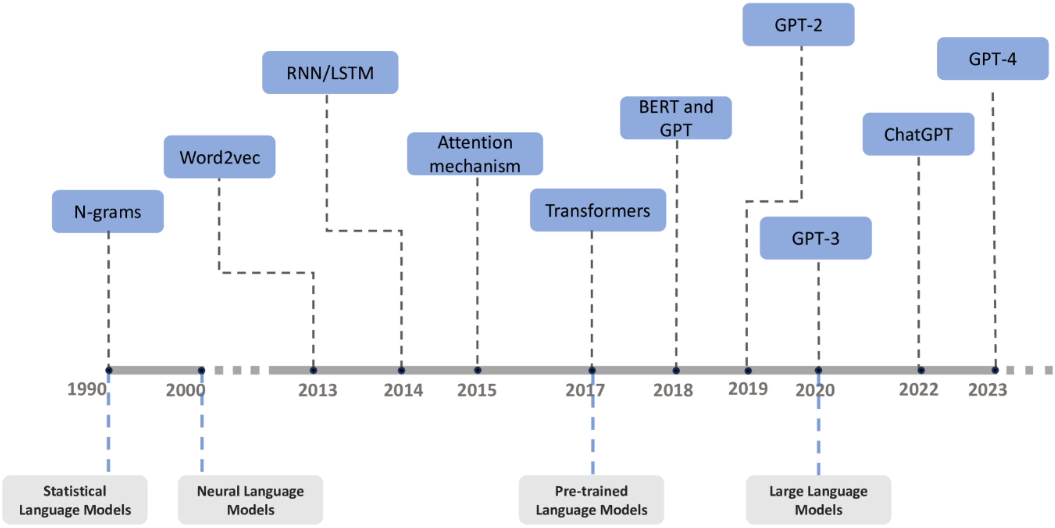
## Lecture 21: Scaling Laws and Instruction Tuning

**Mirella Lapata**
School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

# Recap

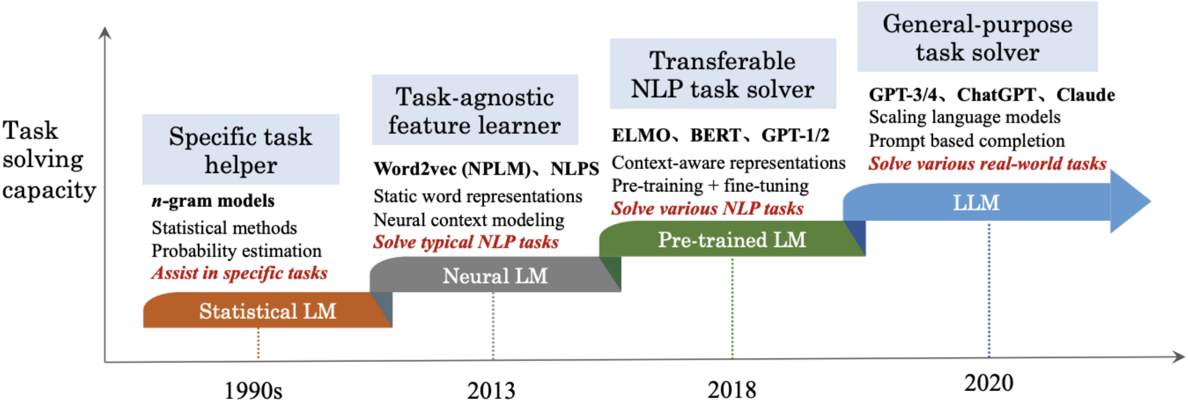
- GPT-3 has arrived! Ginormous LLM.
- *In-context learning* allows us to perform inference with LLMs without fine-tuning.
- Accuracy is highly sensitive to *prompt design*.
- Conclusion so far: *bigger is better*.
- **Today:** how does pretraining a model like GPT-3 actually work? And are we really done with fine-tuning?

# Where are we?

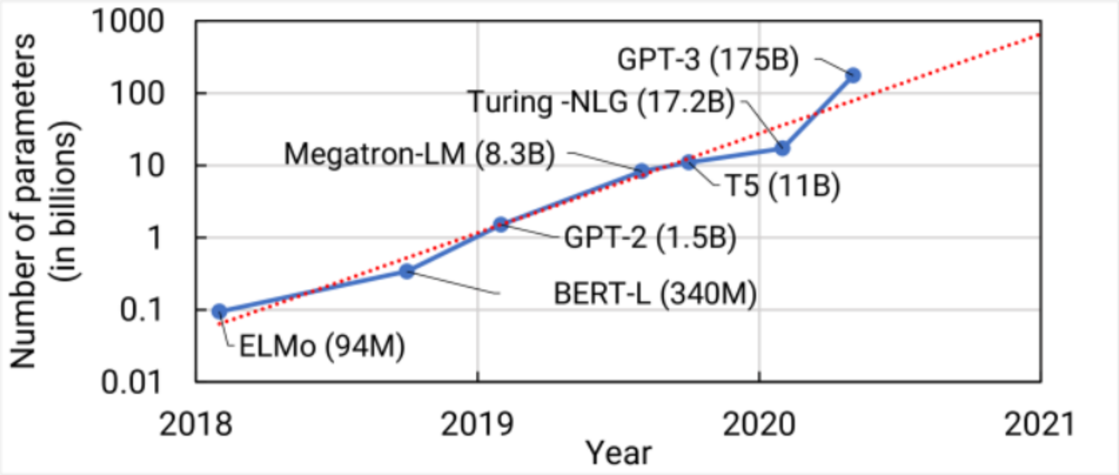


GPT-2
GPT-4
RNN/LSTM
BERT and GPT
ChatGPT
Word2vec
Attention mechanism
N-grams
Transformers
GPT-3
1990
2000
2013
2014
2015
2017
2018
2019
2020
2022
2023
Statistical Language Models
Neural Language Models
Pre-trained Language Models
Large Language Models

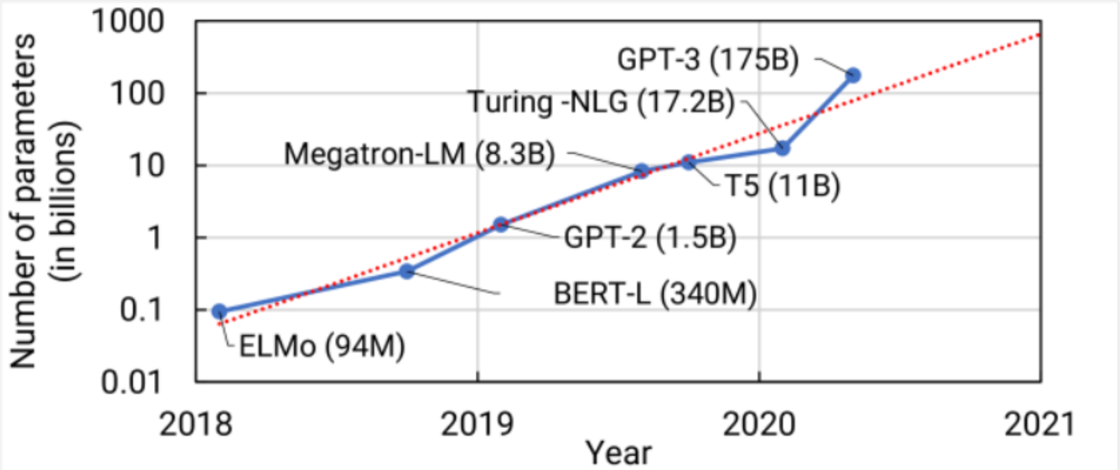
# Where are we?


Task solving capacity
Specific task helper
*n*-gram models
Statistical methods
Probability estimation
Assist in specific tasks
Statistical LM
Task-agnostic feature learner
Word2vec (NPLM)、NLPS
Static word representations
Neural context modeling
Solve typical NLP tasks
Neural LM
Transferable NLP task solver
ELMO、BERT、GPT-1/2
Context-aware representations
Pre-training + fine-tuning
Solve various NLP tasks
Pre-trained LM
General-purpose task solver
GPT-3/4、ChatGPT、Claude
Scaling language models
Prompt based completion
Solve various real-world tasks
LLM
1990s
2013
2018
2020

# Models for language have become bigger



# Models for language have become bigger




Bigger models mean that training and deployment is expensive!

# What does scaling mean?

- Scaling is *not just* about models with more parameters
- Scaling is about using *more compute*:
  (1) More compute for model forward and backward passes
  (2) More compute for training iterations also
- But also about *model capacity*: only a large enough model can take advantage of the additional training

# Larger models present new problems

- We *cannot find the best hyperparameters* by training multiple models.
- We don't know when to stop training!
- Given a *budget for compute*, should we increase the *model size* or the *number of training steps* using that budget
- Can we develop a theory that connects loss with the model sizes and the number of training steps?

# How big should your model be?

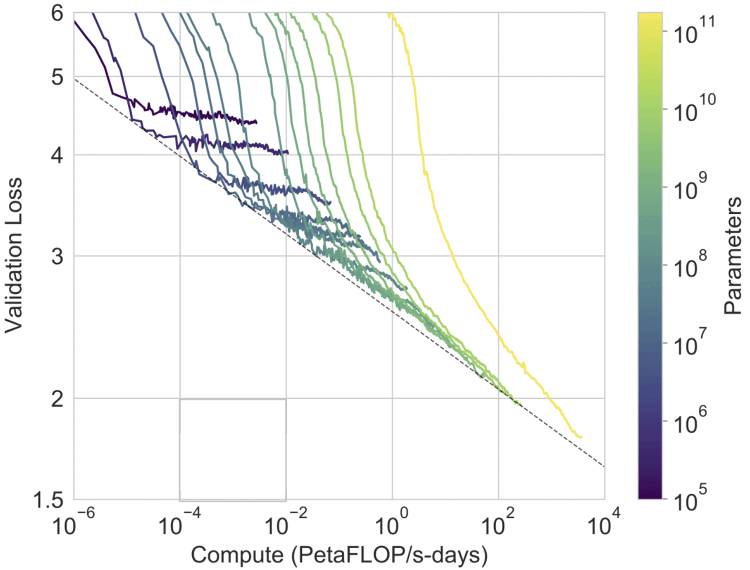
**Petaflop-s-days measure**

Is a measure of **computational capacity** used to quantify the processing power over time, combining performance (measured in petaflops) and the duration (measured in days). One petaflop is equal to **$10^{15}$ floating-point** operations per second.

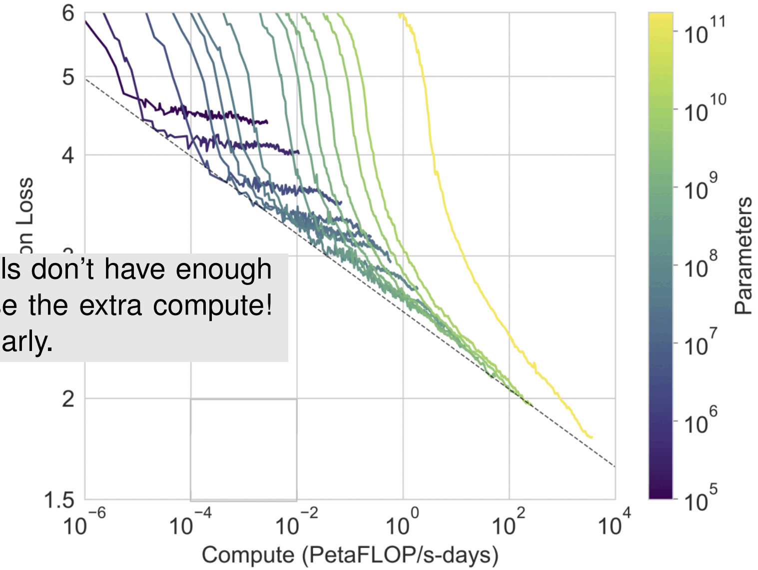
Imagine a supercomputer that has a *processing speed* of **1 petaflop**:

- If this computer runs continuously for **24 hours (1 day)**, it will perform:
  1 petaflop $\times$ 86,400 seconds (in a day) $=$ 8.64 $\times$ $10^{19}$ operations in one day
- If a faster 10-petaflop supercomputer runs for 5 days, it will achieve:
  10 petaflops $\times$ 5 days $=$ 50 petaflop-s-days.
- Training a large language model might require 10,000 petaflop-s-days, meaning it needs to run on a supercomputer with 100 petaflops for 100 days.

# How big should your model be?

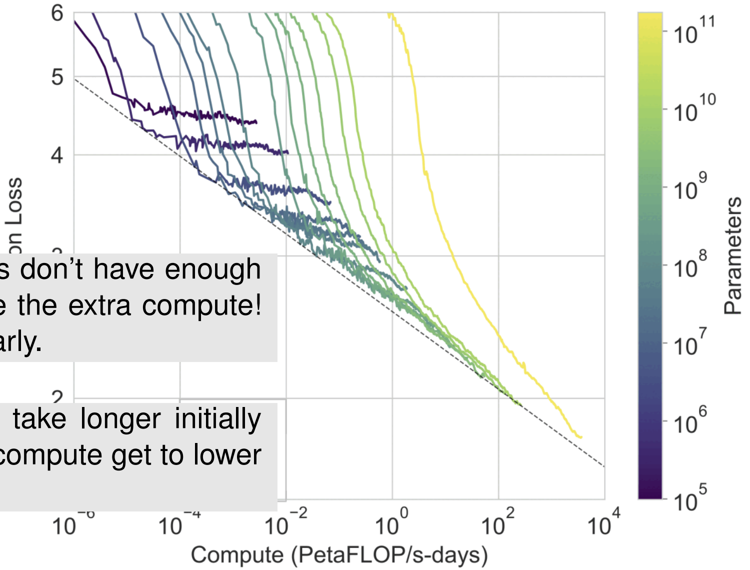


# How big should your model be?




Smaller models don't have enough capacity to use the extra compute! they plateau early.

# How big should your model be?




Smaller models don’t have enough capacity to use the extra compute! they plateau early.

Larger models take longer initially but with more compute get to lower losses.

# How big should your model be?

- For a given compute budget, what is the optimal model size?
- Rather than training models to convergence, train them to optimality.
- But to make this choice, we need to know all these learning curves.
- How can we get them **without** training a model?
- Or when the budget only allows training one LARGE model?

**The claim**

Test loss is a **power law function** of model size and compute. If this is true, then use small models to fit the constants of the power law function, and then extrapolate to large sizes.

# Scaling laws according to Kaplan et al.

Language modeling performance improves smoothly as we increase the **model size**, **dataset size**, and **amount of compute** used for training.

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N} \quad L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D} \quad L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

- Empirical performance has a power-law relationship with each individual factor when the other two properties are held constant.

# Scaling laws according to Kaplan et al.

Language modeling performance improves smoothly as we increase the **model size**, **dataset size**, and **amount of compute** used for training.

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N} \quad L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D} \quad L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

- N: number of model parameters (no token embeddings, positional embeddings), $D$: dataset size, $C$: compute budget
- Empirical performance has a power-law relationship with each individual factor when the other two properties are held constant.

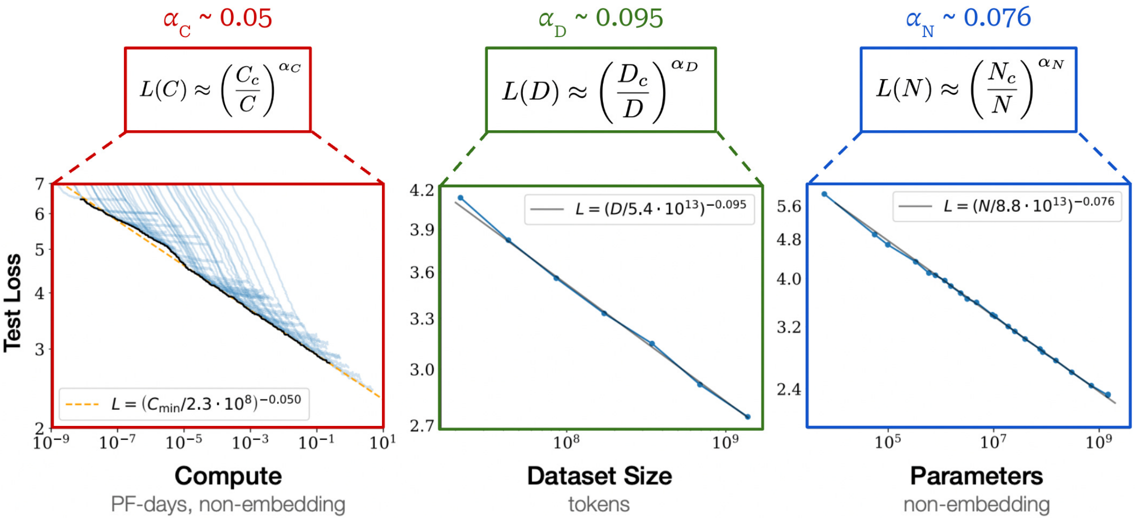
# Scaling laws according to Kaplan et al.

Language modeling performance improves smoothly as we increase the **model size**, **dataset size**, and **amount of compute** used for training.

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N} \quad L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D} \quad L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

- N: number of model parameters (no token embeddings, positional embeddings), $D$: dataset size, $C$: compute budget
- $N_C$, $\alpha_N$, $D_c$, $C_c$, $\alpha_N$, $\alpha_D$, $\alpha_C$, depend on exact transformer architecture.
- Empirical performance has a power-law relationship with each individual factor when the other two properties are held constant.

# Scaling laws according to Kaplan et al.




LLMs pretrained up to 1.5B parameters over subsets of WebText2 corpus (22M to 23B tokens), fixed context length of 1,024 tokens and next-token prediction loss.

# Working out the parameters of GPT-style Transformer

The number of (non-embedding) parameters $N$ can be roughly computed as follows:

$$
\begin{aligned}
N &\approx 2\ d\ n_{\text{layer}}(2\ d_{\text{attn}} + d_{\text{ff}}) \\
&\approx 12\ n_{\text{layer}}\ d^2 \\
&\quad (\text{assuming } d_{\text{attn}} = d_{\text{ff}}/4 = d)
\end{aligned}
$$

- We are ignoring biases, $d$ is the input and output dimensionality of the model
- $d_{\text{attn}}$ as the self-attention layer size, $d$ff the size of the feedforward layer
- GPT-3: $n = 96$ layers, $d = 12288$.
- It has $12 \times 96 \times 12288^2 \approx 175$ billion parameters.

# Empirical Observations

- For given amount of compute $C$, *best loss* we can obtain is:

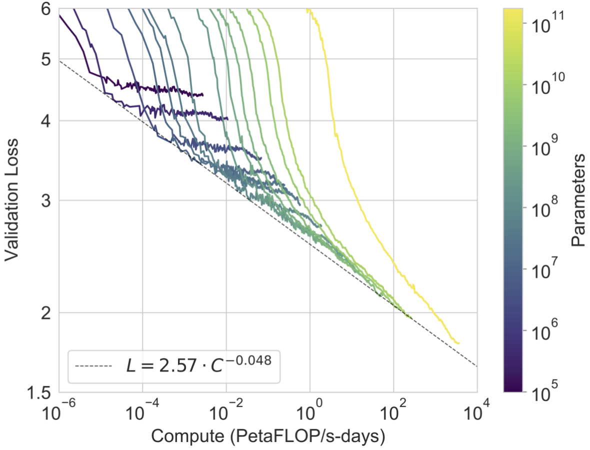
$$L \propto C^{-0.048}$$

- For given amount of compute $C$, *optimal model size* is:

$$N_{opt} \propto C^{0.73}$$

- For given amount of compute $C$, *optimal number of tokens*:

$$D_{opt} \propto C^{0.27}$$



# Empirical Observations

- For given amount of compute $C$, *best loss* we can obtain is:

$$L \propto C^{-0.048}$$

- For given amount of compute $C$, *optimal model size* is:

$$N_{opt} \propto C^{0.73}$$

- For given amount of compute $C$, *optimal number of tokens*:

$$D_{opt} \propto C^{0.27}$$

Suppose we have access to 100x more compute. What should we do, increase the model size or the number of tokens?

# Empirical Observations

- For given amount of compute $C$, *best loss* we can obtain is:

$$L \propto C^{-0.048}$$

- For given amount of compute $C$, *optimal model size* is:

$$N_{opt} \propto C^{0.73}$$

- For given amount of compute $C$, *optimal number of tokens*:

$$D_{opt} \propto C^{0.27}$$

Suppose we have access to 100x more compute. What should we do, increase the model size or the number of tokens?

$$\textit{new } N_{opt} \propto (100C)^{0.73}$$

$$\textit{new } D_{opt} \propto (100C)^{0.27}$$

# Empirical Observations

- For given amount of compute $C$, *best loss* we can obtain is:

$$L \propto C^{-0.048}$$

- For given amount of compute $C$, *optimal model size* is:

$$N_{opt} \propto C^{0.73}$$

- For given amount of compute $C$, *optimal number of tokens*:

$$D_{opt} \propto C^{0.27}$$

Suppose we have access to 100x more compute. What should we do, increase the model size or the number of tokens?

$$\textit{new } N_{opt} \propto (100C)^{0.73}$$

$$\textit{new } D_{opt} \propto (100C)^{0.27}$$

$$\frac{\textit{new } N_{opt}}{\textit{old } N_{opt}} = \frac{(100C)^{0.73}}{C^{0.73}} = 100^{0.73} \approx 28.8$$

$$\frac{\textit{new } D_{opt}}{\textit{old } D_{opt}} = \frac{(100C)^{0.27}}{C^{0.27}} = 100^{0.27} \approx 3.47$$

# Empirical Observations

- For given amount of compute $C$, *best loss* we can obtain is:

$$L \propto C^{-0.048}$$

- For given amount of compute $C$, *optimal model size* is:

$$N_{opt} \propto C^{0.73}$$

- For given amount of compute $C$, *optimal number of tokens*:

$$D_{opt} \propto C^{0.27}$$

Suppose we have access to 100x more compute. What should we do, increase the model size or the number of tokens?

$$\textit{new } N_{opt} \propto (100C)^{0.73}$$

$$\textit{new } D_{opt} \propto (100C)^{0.27}$$

$$\frac{\textit{new } N_{opt}}{\textit{old } N_{opt}} = \frac{(100C)^{0.73}}{C^{0.73}} = 100^{0.73} \approx 28.8$$

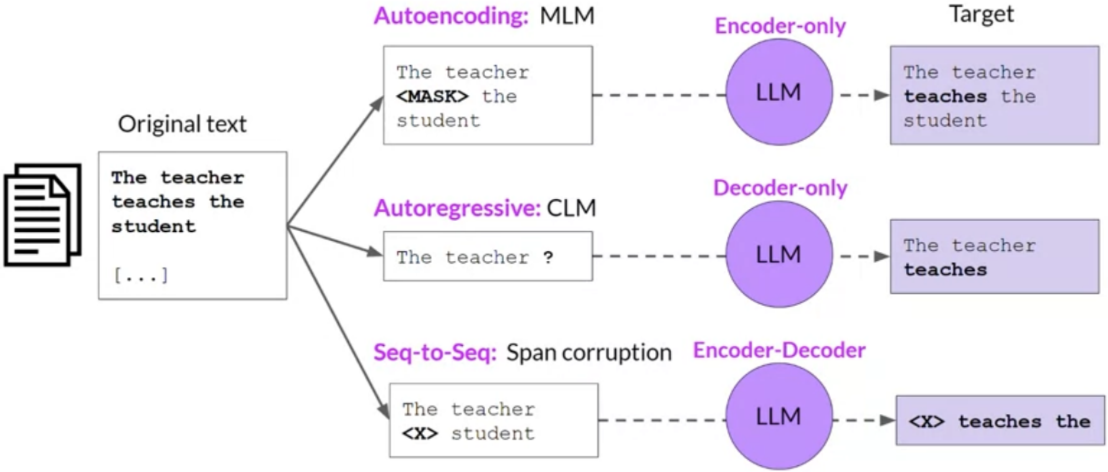
$$\frac{\textit{new } D_{opt}}{\textit{old } D_{opt}} = \frac{(100C)^{0.27}}{C^{0.27}} = 100^{0.27} \approx 3.47$$

Increase number of training steps by 29x and number of tokens by 3.5x. *We estimate these without training the model!*

# Interim Summary

- Scaling laws are empirical observations that relate model size, compute in FLOPs, training size and loss functions.
- Typically power law relationships.
- These are empirical observations. There is very little theoretical understanding.
- Nevertheless, very useful to train large models without trial and error.

# Model architectures and pretraining objectives



# Is language modeling the same as following instructions?

Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence

GPT-3 175B

Serendipity is the ability to see something good in something bad. Use the word in a sentence.

Serendipity means to be happy with what you have. Use the word in a sentence.

Serendipity means finding something good in something that is bad. Use the word in a sentence.

# Is language modeling the same as following instructions?

GPT-3 175B

Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence

Serendipity is the ability to see something good in something bad. Use the word in a sentence.

Serendipity means to be happy with what you have. Use the word in a sentence.

Serendipity means finding something good in something that is bad. Use the word in a sentence.

What is a reasonable answer to this prompt?

# Is language modeling the same as following instructions?

GPT-3 175B

Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence

Serendipity is the ability to see something good in something bad. Use the word in a sentence.

Serendipity means to be happy with what you have. Use the word in a sentence.

Serendipity means finding something good in something that is bad. Use the word in a sentence.

What is a reasonable answer to this prompt?

*Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.*

# What we’ve seen so far

## (A) Pretrain-finetune (BERT, T5)

**Pretrained LM** → **Finetune on task A** → **Inference on task A**

- Typically requires many task-specific examples.
- One specialized model for each task.

## (B) Prompting (GPT3)

**Pretrained LM** → **Inference on task A**

- General-purpose model.
- Specialize via few-shot prompting or prompt engineering.

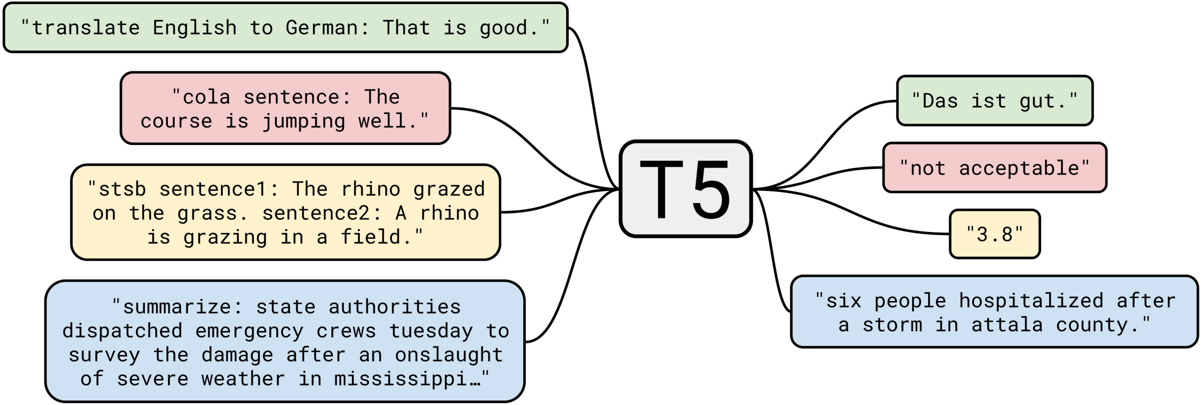
# Instruction Tuning

## (C) Instruction tuning

**Pretrained LM** → **Instruction-tune on many tasks: B, C, D,** → **Inference on task A**

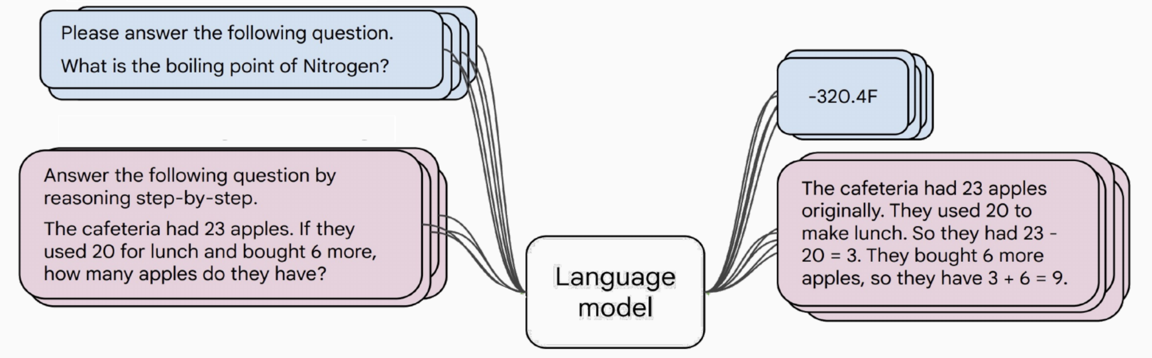
- Model learns many tasks via natural language instructions.
- Inference on unseen tasks!

# Do you remember T5?



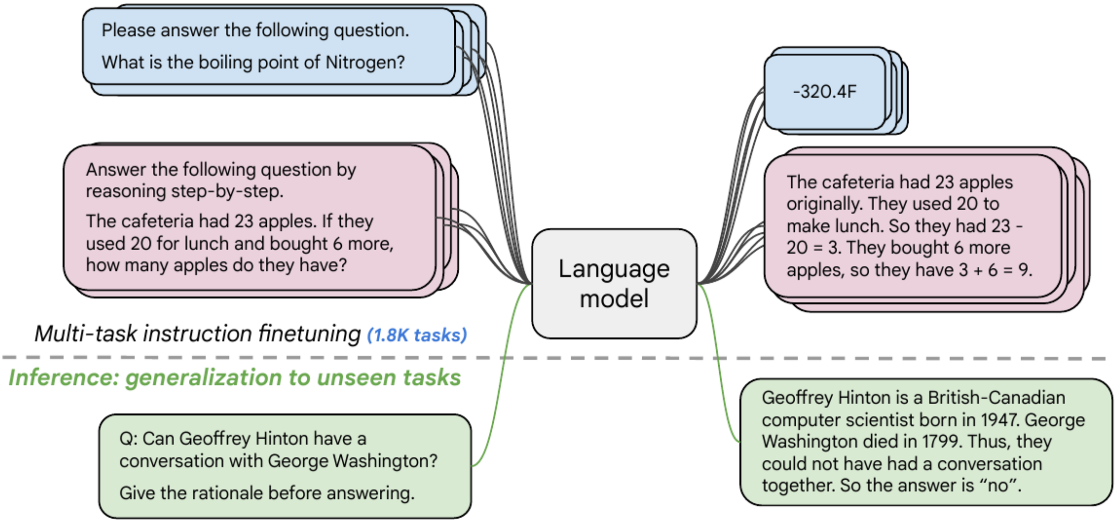

For each task, design a template so that the inputs and outputs are text.

# Instruction finetuning

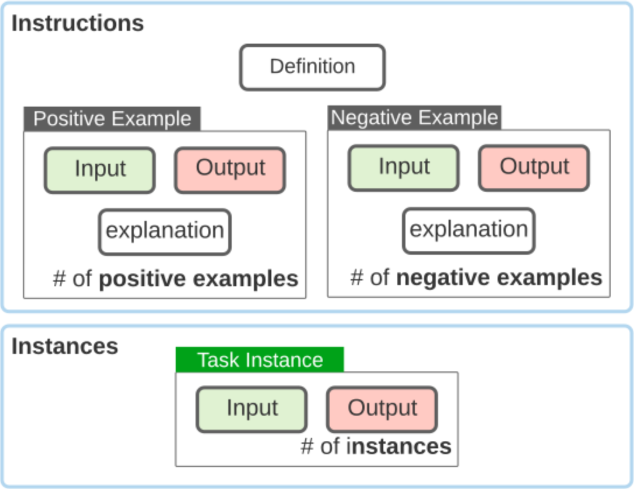



Inputs and outputs are both text. The output *is not a completion* of the input text (as with the language modeling objective), but *the response* to it.

# Instruction finetuning



# Natural Instructions




- Humans can solve different tasks, by simply reading instructions and looking at a few examples.
- NATURAL INSTRUCTIONS has 61 distinct tasks, their instructions, and 193k task instances (input-output pairs).
- Using meta-dataset, we can train models on *seen* tasks and measure generalization on *unseen* ones.

Explore instructions `https://instructions.apps.allenai.org/`.

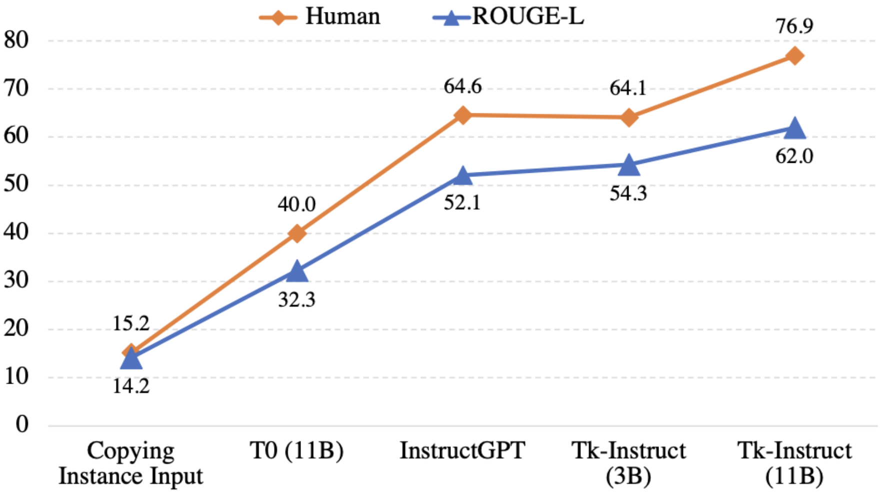
# Super-Natural Instructions


- Super-NaturalInstructions dataset contains over 1.6K tasks, 3M+ examples
- Classification, sequence tagging, rewriting, translation, QA
- Many languages: 576 non-English

# Super-Natural Instructions Example

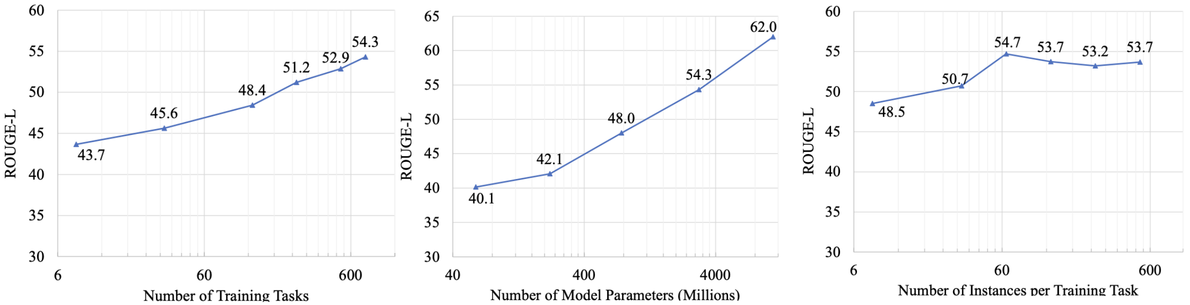
| | |
|---|---|
| Task Type | Textual Entailment |
| Task ID | task1344_rte_textual_entailment |
| Definition | In this task, you're given two sentences. Indicate if the first sentence clearly entails the second sentence (i.e., one can conclude the 2nd sentence by reading the 1st one). Indicate your answer with "1" if the first sentence entails the second sentence, otherwise answer with "0". |
| Positive Example | **Input**: Sentence 1: No Weapons of Mass Destruction Found in Iraq Yet. Sentence 2:Weapons of Mass Destruction Found in Iraq.<br>**Output**: 0<br>**Explanation**: In our first statement we clearly say that Iraq does not have any weapon of mass destruction but the second sentence says that weapon of mass destruction is found in Iraq which is a contradiction. Hence output will be 0 for non entailment. |
| Negative Example | **Input**: Sentence 1: Valero Energy Corp., on Monday, said it found "extensive" additional damage at its 250,000-barrel-per-day Port Arthur refinery. Sentence 2: Valero Energy Corp. produces 250,000 barrels per day.<br>**Output**: 0<br>**Explanation**: The first statement mentions that there was damage found in the 250,000 barrel-per-day Port Aurthur refinery. Which means that they produce 250,000 barrels a day. Hence the output should have been 1 for entailment. |
| Instance | **Input**: Sentence 1: Like the United States, U.N. officials are also dismayed that Aristide killed a conference called by Prime Minister Robert Malval in Port-au-Prince in hopes of bringing all the feuding parties together. Sentence 2: Aristide had Prime Minister Robert Malval murdered in Port-au-Prince.<br>**Valid Output**: ["0"] |

# Super-Natural Instructions Results

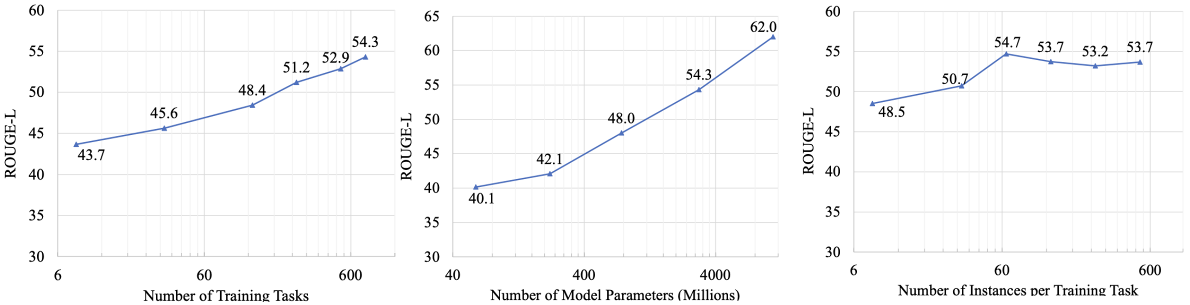



Models that leverage instructions show stronger generalization to unseen tasks.

# Scaling Instruction-tuning

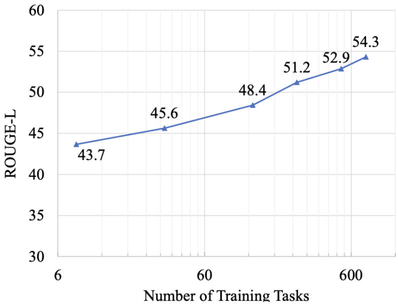


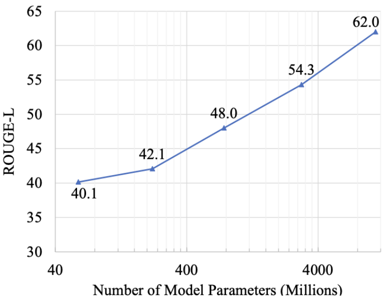
# Scaling Instruction-tuning

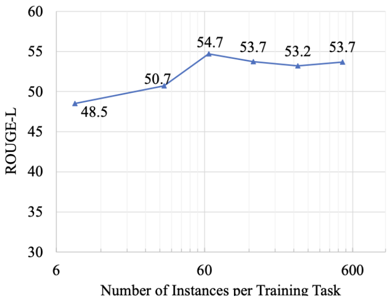



Linear growth of model performance with exponential increase in observed tasks and model size

# Scaling Instruction-tuning



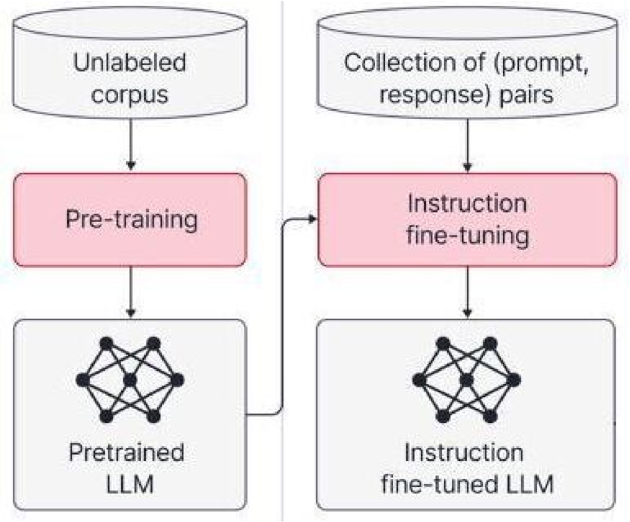




Linear growth of model performance with exponential increase in observed tasks and model size

Number of examples has little effect

# Training Large Language Models




- What just happened?
- Finetuning is back!
- We are fine-tuning on *instructions* rather creating different copies of the same model for different tasks.

## Summary

- Pre-trained LLMs *actually fail* to properly follow the prompts.
- A simple strategy to address this *instruction tuning*.
- The LLM is trained on a small dataset of examples that consists of prompts or instructions followed by the correct actions.
- By *fine-tuning* on these examples (usually very few per task), the model learns to better understand and follow instructions in natural language.
- An instruction-tuned LLM will often be able to *generalize* and follow instructions on a much wider variety of tasks.
- Still not a great solution for open-ended tasks (e.g., “Write me a story about a dog and her pet grasshopper.”).

**Next time:** Reinforcement learning from human feedback.