Large Language Models

Generative AI and Foundation Models

Spring 2024

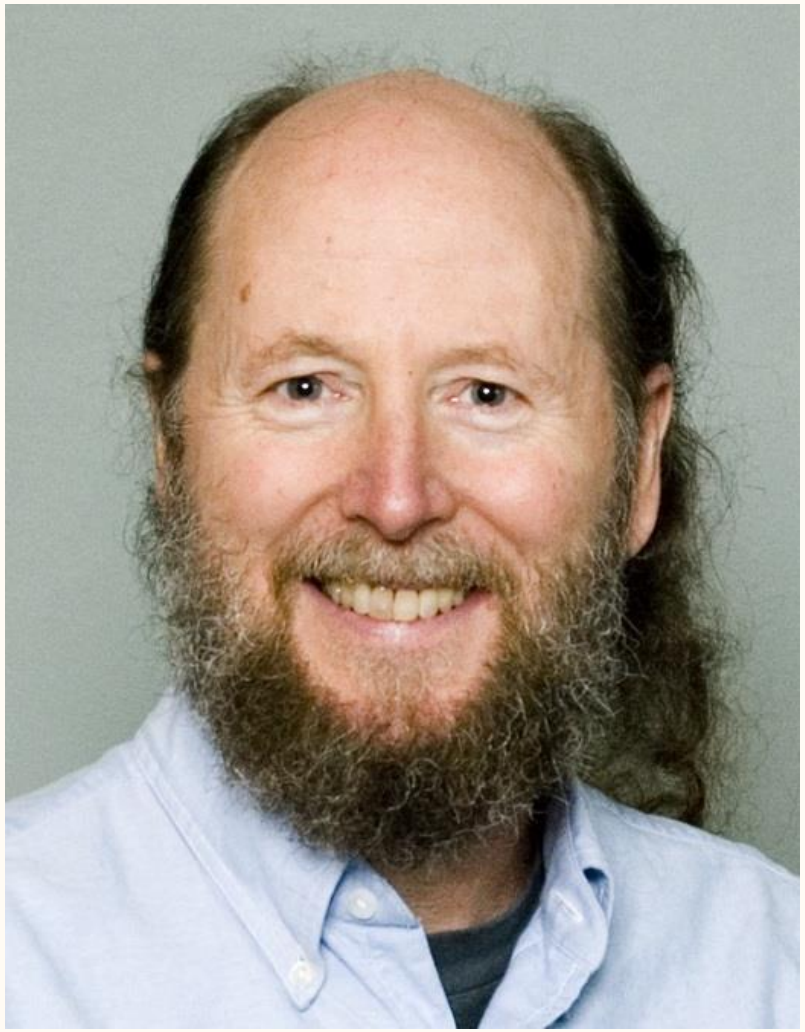
Department of Mathematical Sciences

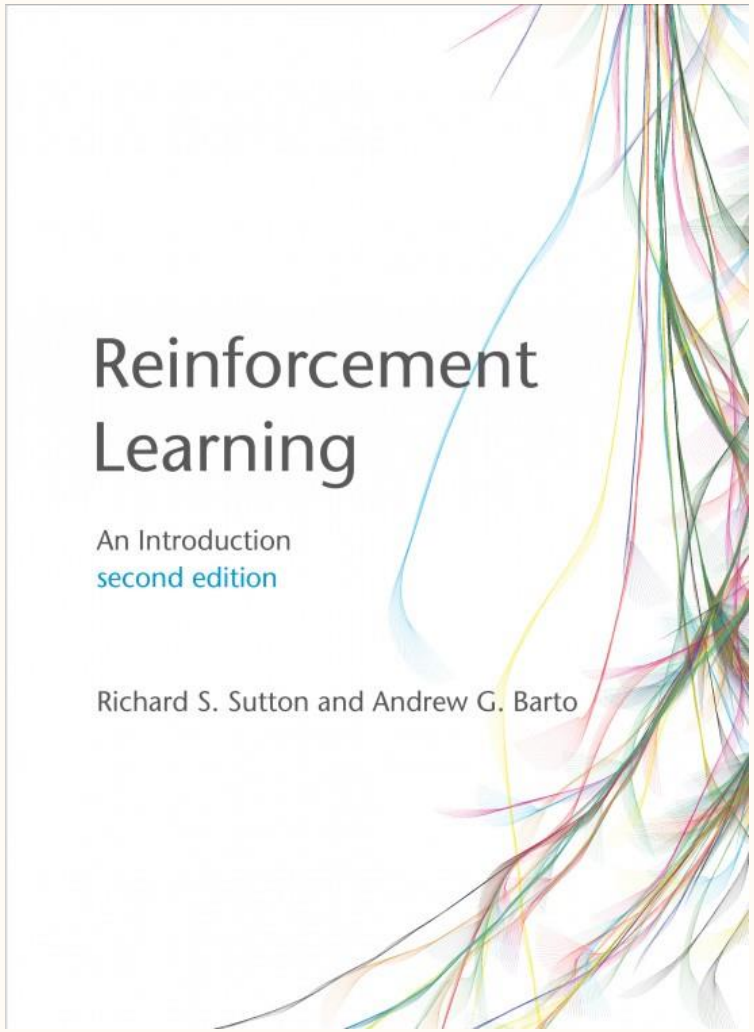
Ernest K. Ryu

Seoul National University

Richard M. Sutton

One of the founding fathers of reinforcement learning.





The Bitter Lesson

“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 one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent."

The 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. When a simpler, search-based approach with special hardware and software proved vastly more effective, these human-knowledge-based chess researchers were not good losers. They said that “brute force” search may have won this time, but it was not a general strategy, and anyway it was not how people played chess. These researchers wanted methods based on human input to win and were disappointed when they did not.”

The Bitter Lesson

“A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning by self play, and learning in general, is like search in that it enables massive computation to be brought to bear. Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research. In computer Go, as in computer chess, researchers' initial effort was directed towards utilizing human understanding (so that less search was needed) and only much later was much greater success had by embracing search and learning.”

The Bitter Lesson

“In speech recognition, there was an early competition, sponsored by DARPA, in the 1970s. Entrants included a host of special methods that took advantage of human knowledge—knowledge of words, of phonemes, of the human vocal tract, etc. On the other side were newer methods that were more statistical in nature and did much more computation, based on hidden Markov models (HMMs). Again, the statistical methods won out over the human-knowledge-based methods. This led to a major change in all of natural language processing, gradually over decades, where statistics and computation came to dominate the field. The recent rise of deep learning in speech recognition is the most recent step in this consistent direction. Deep learning methods rely even less on human knowledge, and use even more computation, together with learning on huge training sets, to produce dramatically better speech recognition systems. As in the games, researchers always tried to make systems that worked the way the researchers thought their own minds worked — they tried to put that knowledge in their systems — but it proved ultimately counterproductive, and a colossal waste of researcher's time, when, through Moore's law, massive computation became available and a means was found to put it to good use."

The Bitter Lesson

“In computer vision, there has been a similar pattern. Early methods conceived of vision as searching for edges, or generalized cylinders, or in terms of SIFT features. But today all this is discarded. Modern deep-learning neural networks use only the notions of convolution and certain kinds of invariances, and perform much better.”

The Bitter Lesson

“This is a big lesson. As a field, we still have not thoroughly learned it, as we are continuing to make the same kind of mistakes. To see this, and to effectively resist it, we have to understand the appeal of these mistakes. We have to learn the bitter lesson that building in how we think we think does not work in the long run. The bitter lesson is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning. The eventual success is tinged with bitterness, and often incompletely digested, because it is success over a favored, human-centric approach.”

The Bitter Lesson

“One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are search and learning.”

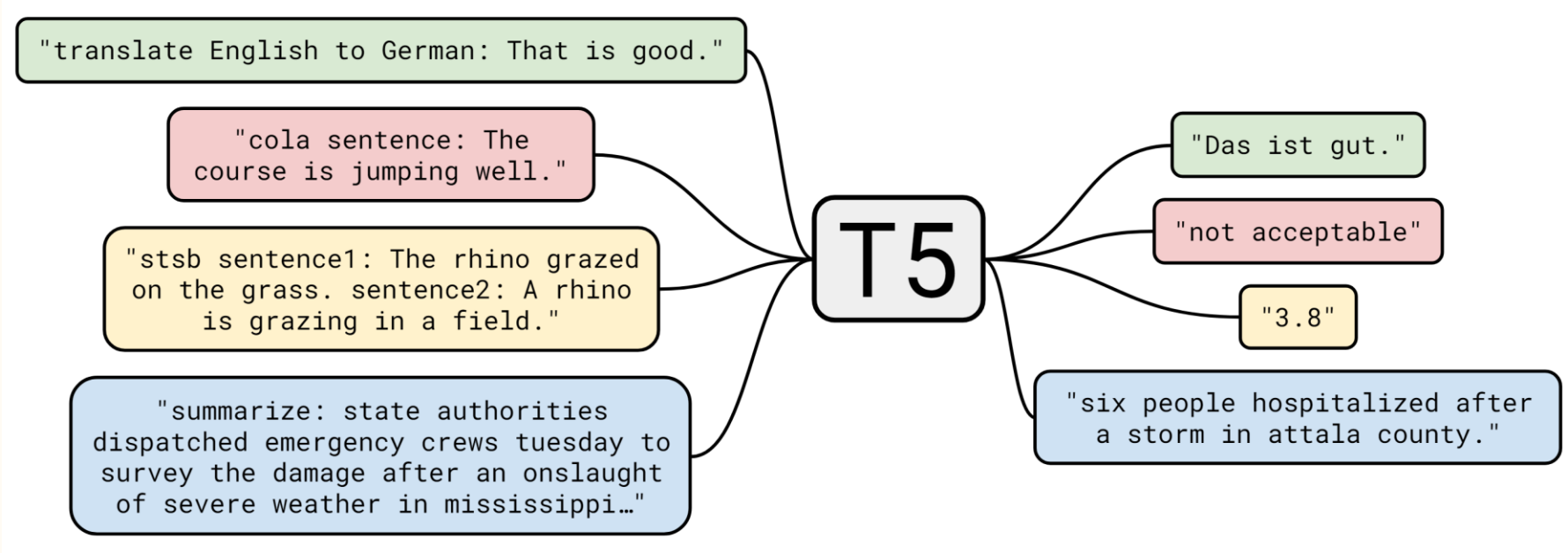
The Bitter Lesson

“The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; instead we should build in only the meta-methods that can find and capture this arbitrary complexity. Essential to these methods is that they can find good approximations, but the search for them should be by our methods, not by us. We want AI agents that can discover like we can, not which contain what we have discovered. Building in our discoveries only makes it harder to see how the discovering process can be done.”

— Richard M. Sutton —

March 13, 2019

T5 model



Text-to-text-transfer-transformer (T5) uses an encoder-decoder transformer and formats all pre-training and fine-tuning into a text-to-text format.

Unified task-agnostic architecture. The many tasks, which are not semantically related, are formatted into a text-to-text format. Same model, objective, training procedure and decoding process to every task that we consider.

C. Raffel, N. Shazeer, A. Roberts, K .Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *JMLR*, 2020. (*arXiv* Oct. 2019)

T5 pre-training

Pre-training on large unlabeled text with diverse objectives inspired by prior work.

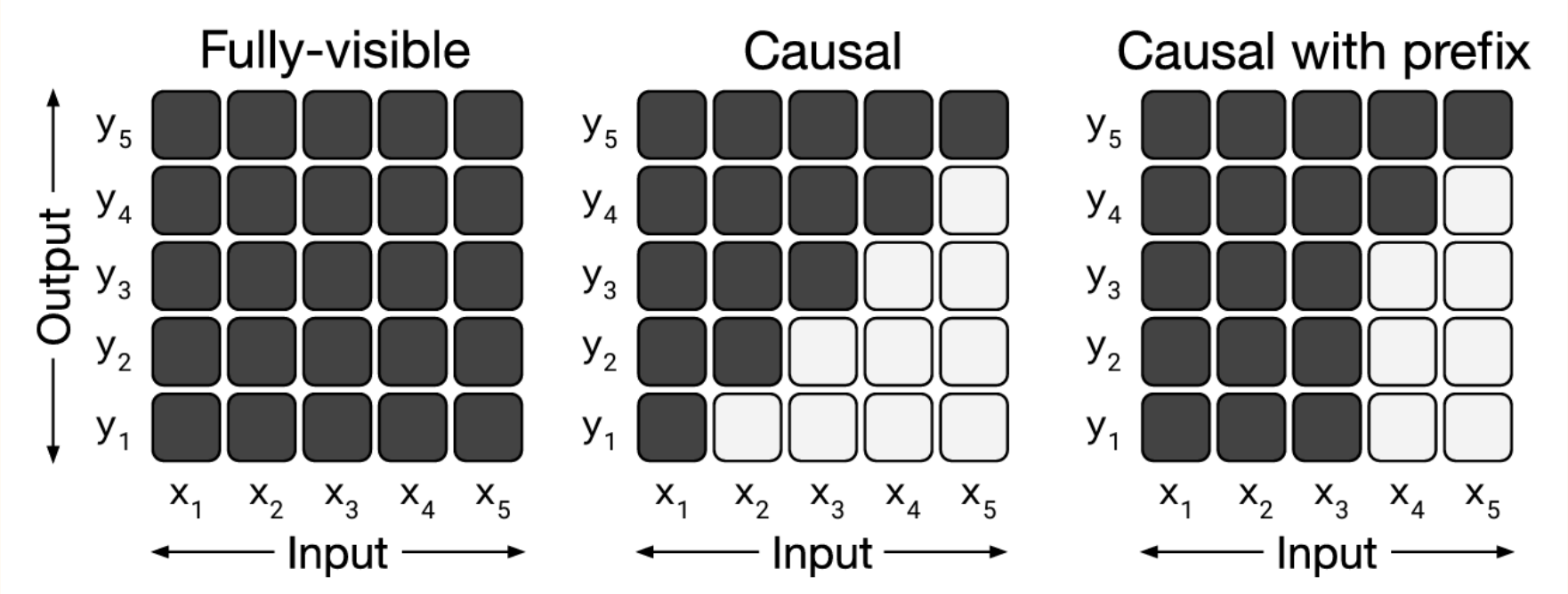
Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

The “inputs” are fed into the encoder block while the “target” text is generated by the decoder one token at a time.

T5 attention mask

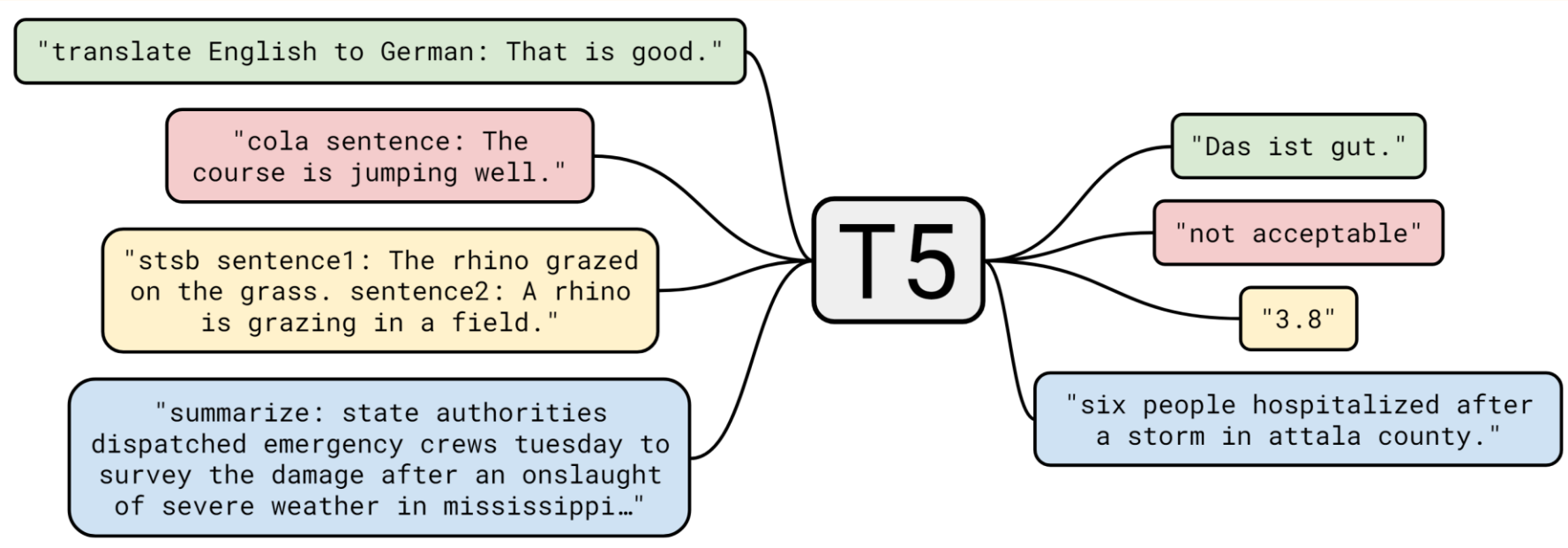
Encoder uses un-masked attention, while decoder can access encoder tokens via cross attention and the earlier decoder tokens.

Alternatively, interpret the T5 transformer as using the “causal with prefix” attention mask.

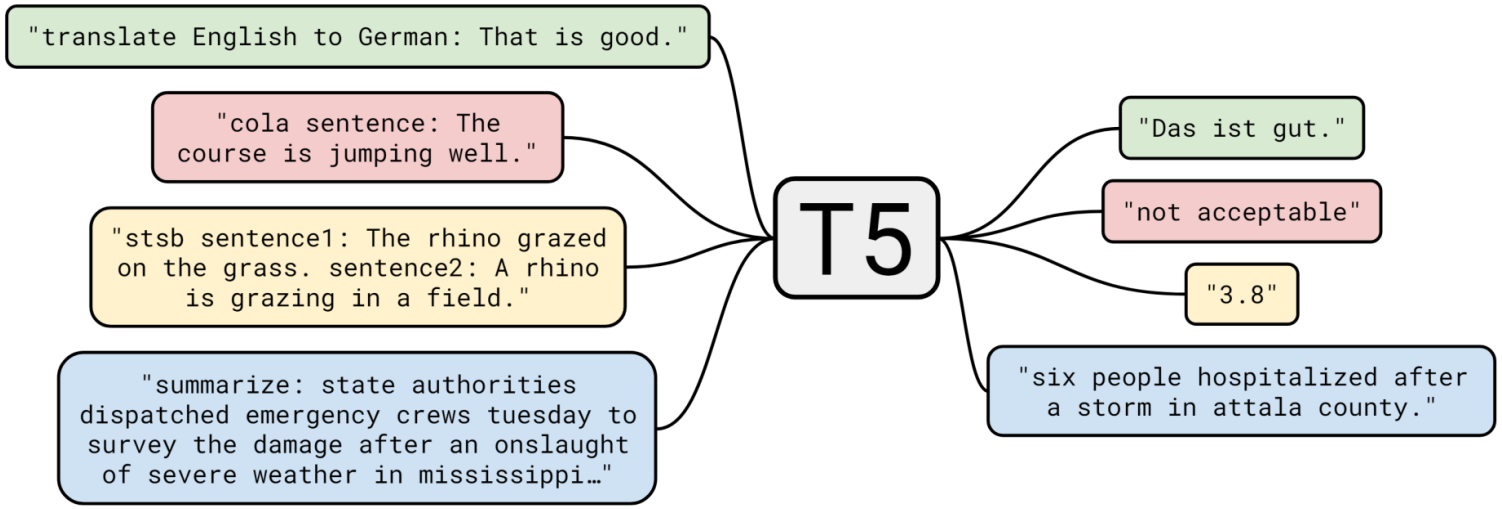


T5 fine-tuning

Simultaneously fine-tune on a wide range of tasks. Simply prompt the model differently for each task to inform T5 of the specific task to solve.



T5 contribution



Advanced state-of-the-art with the pre-train-than-fine-tune approach.

Gave the idea that language models can understand and respond to natural language instructions. We can simply tell a language model what we want (in natural language) and it will follow our instructions.

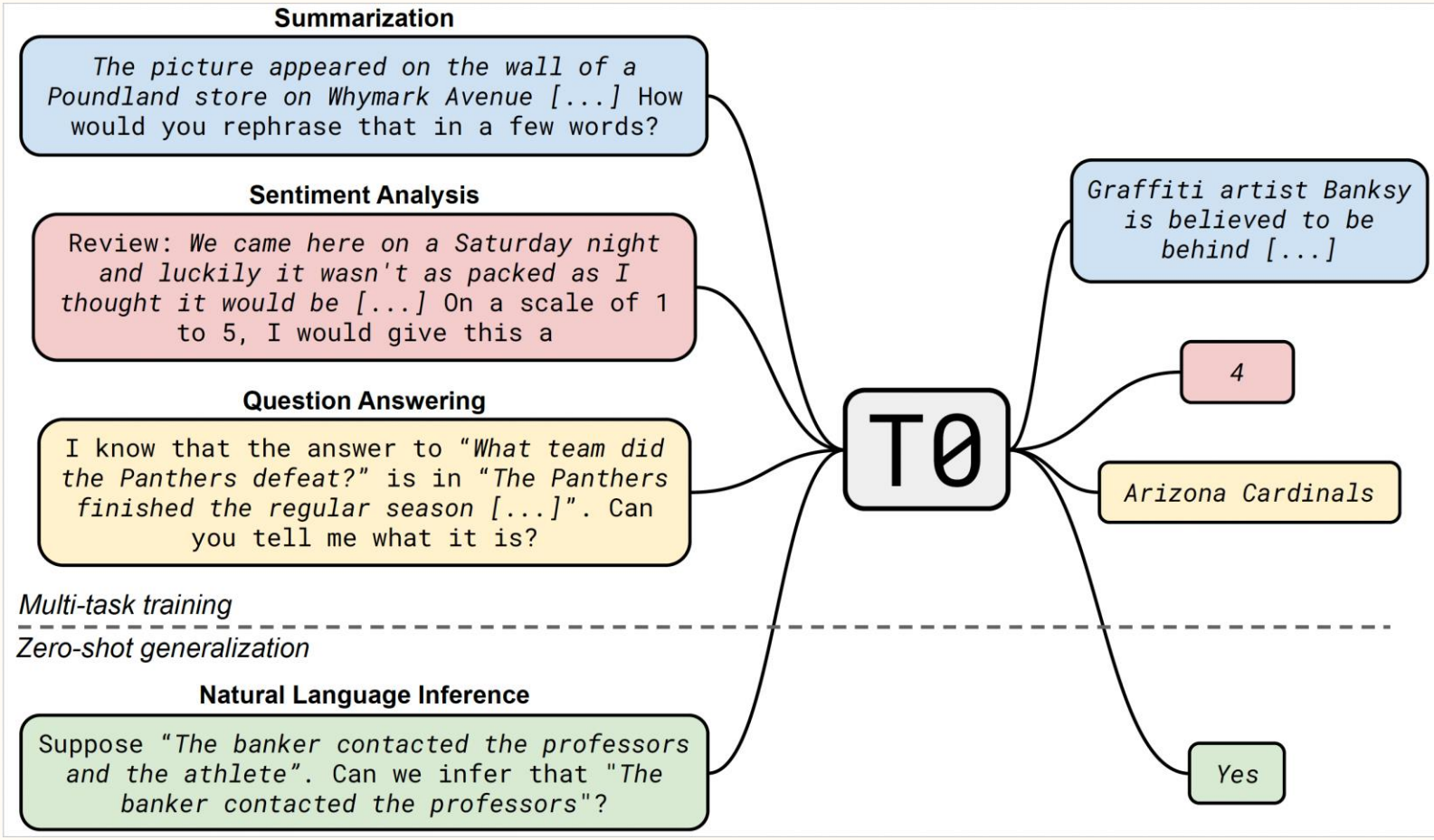
Problem: The prompts were unnatural as they did not fully describe the task at hand.
It was a half-way measure between an arbitrary label (like “task 3A”) and a complete natural-language description.

C. Raffel, N. Shazeer, A. Roberts, K .Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *JMLR*, 2020. (*arXiv* Oct. 2019)

Instruction fine-tuning

Instruction fine-tuning, presented in the FLAN[#] and T0[*] papers, fine-tunes a pre-trained model on a collection of datasets described via natural-language instructions.

This allows the many tasks to be unified: Follow the natural language instructions.

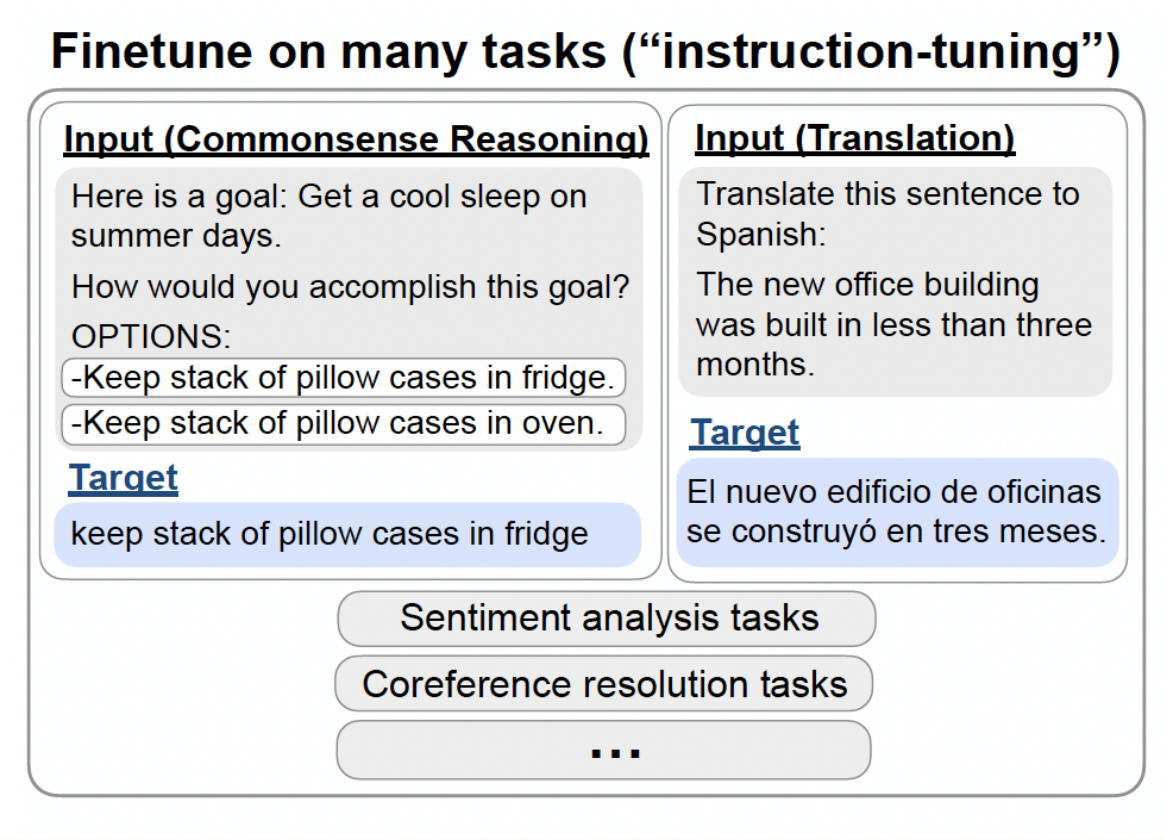


[#]J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned language models are zero-shot learners, *ICLR*, 2022. (*arXiv* Sept. 2021)

[*]V. Sanh, A. Webson, C. Raffel, S. H. Bach, …, Alexander M. Rush, Multitask prompted training enables zero-shot task generalization, *ICLR*, 2022. (*arXiv* Oct. 2021)

Instruction fine-tuning

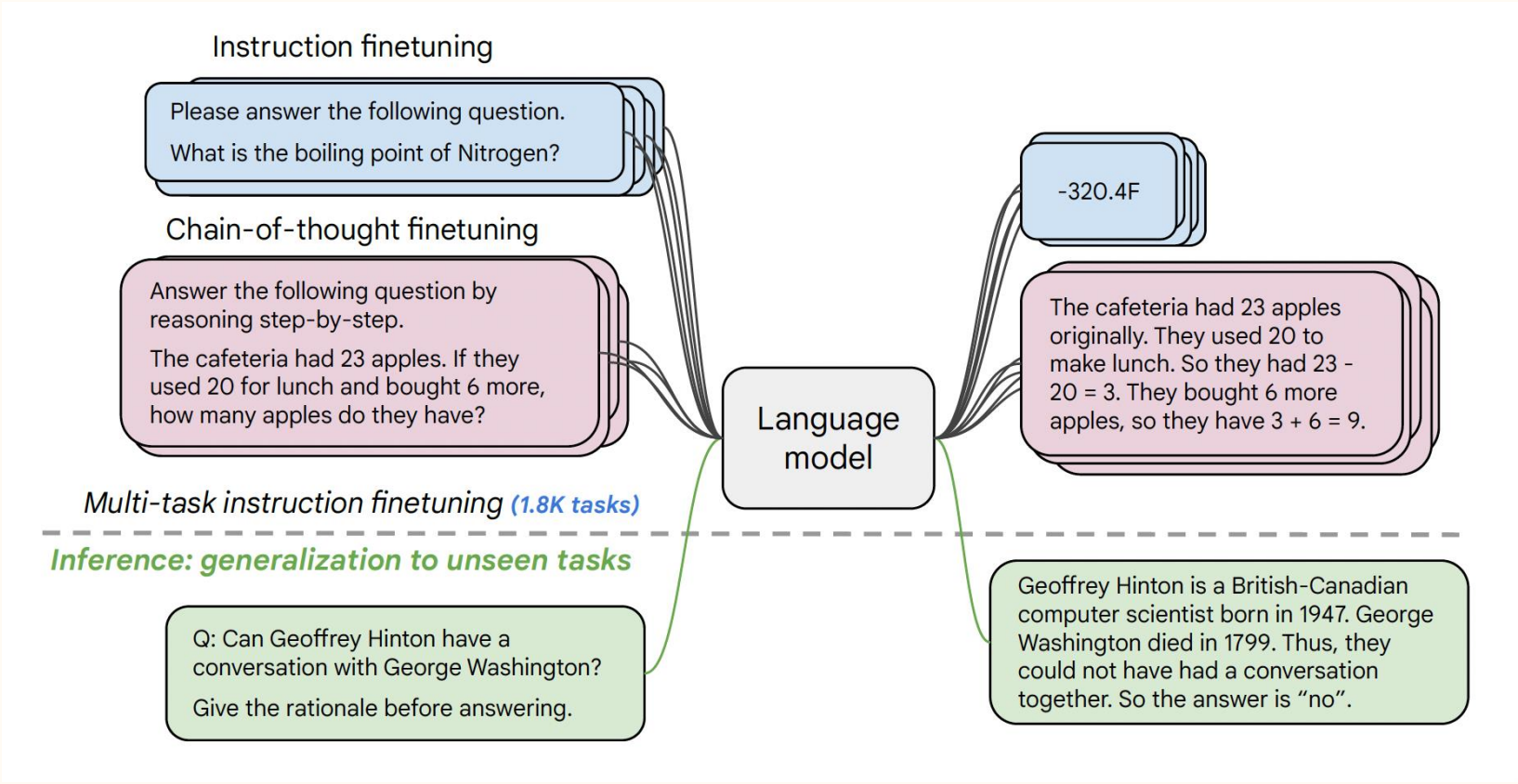
FLAN is a 137B parameter model instruction fine-tuned on over 60 NLP datasets verbalized via natural language instruction templates.



J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned language models are zero-shot learners, *ICLR*, 2022. (*arXiv* Sept. 2021)

Scaling instruction fine-tuning

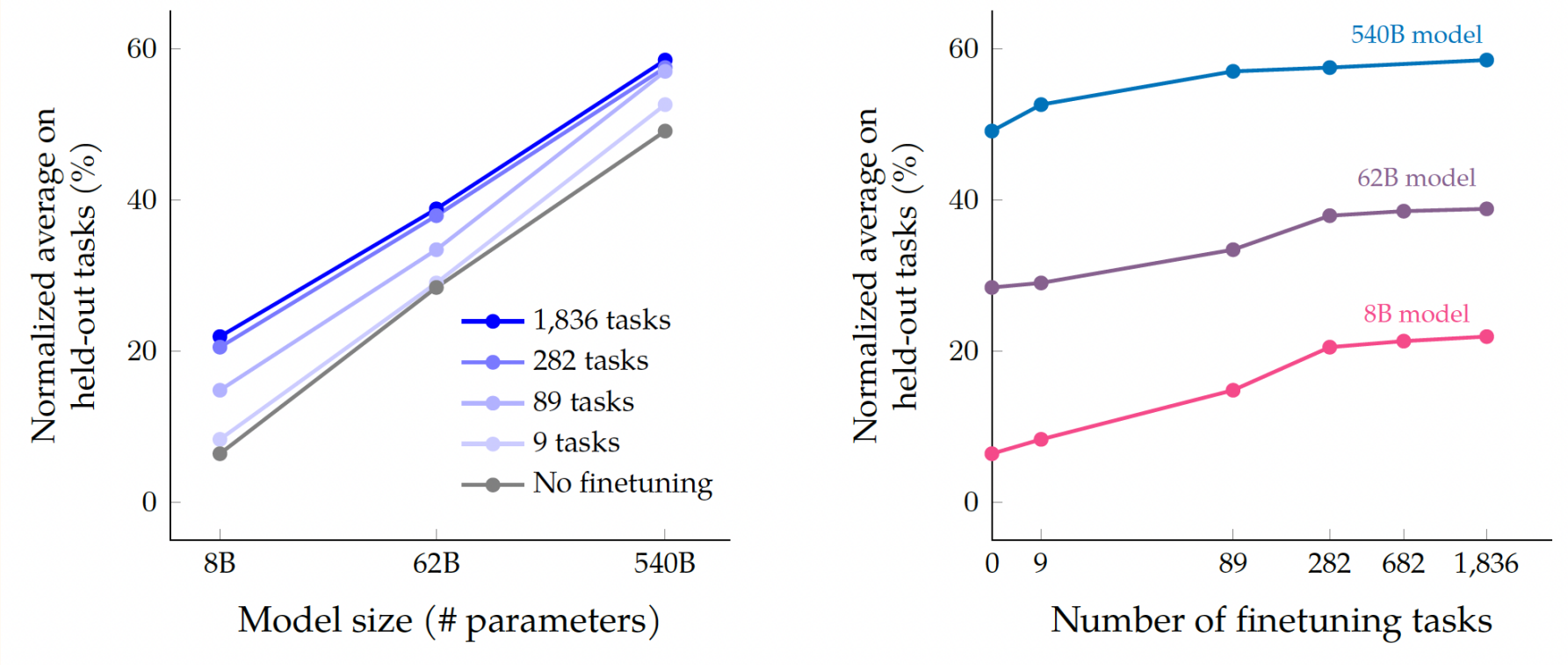
Flan-PaLM scales instruction fine-tuning up to a 540B model with 1836 instruction-finetuning tasks.



H. W. Chung, L. Hou, S. Longpre, … Jason Wei, Scaling instruction-finetuned language models, JMLR, 2024. (arXiv Oct. 2022)

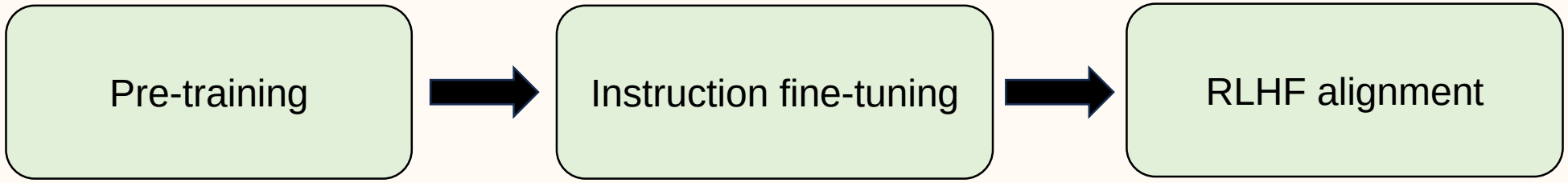
Scaling instruction fine-tuning

Key finding: Task diversity is essential not just in having the model be multi-task, but also in benefiting the individual task performances. Training on tasks A, B, C, … improved performance on task A.



H. W. Chung, L. Hou, S. Longpre, … Jason Wei, Scaling instruction-finetuned language models, JMLR, 2024. (arXiv Oct. 2022)

3-Step of training LLMs



1. Pre-traigning produces model with base capabilties, but the model just tries to complete text. Model does not have the propensity to follow instructions.

2. Instruction fine-tuning induces the model to follow instructions and be helpful.
   Model can engage in chat-bot-style back-and-forth dialogue after instruction fine-tuning.

3. RLHF further aligns LLM with human values and expectations.

Why RLHF?

Pre-training and supervised instruction fine-tuning use the next-token-prediction loss. The dataset presents a correct answer and forces the model to imitate it, like imitation learning of RL.

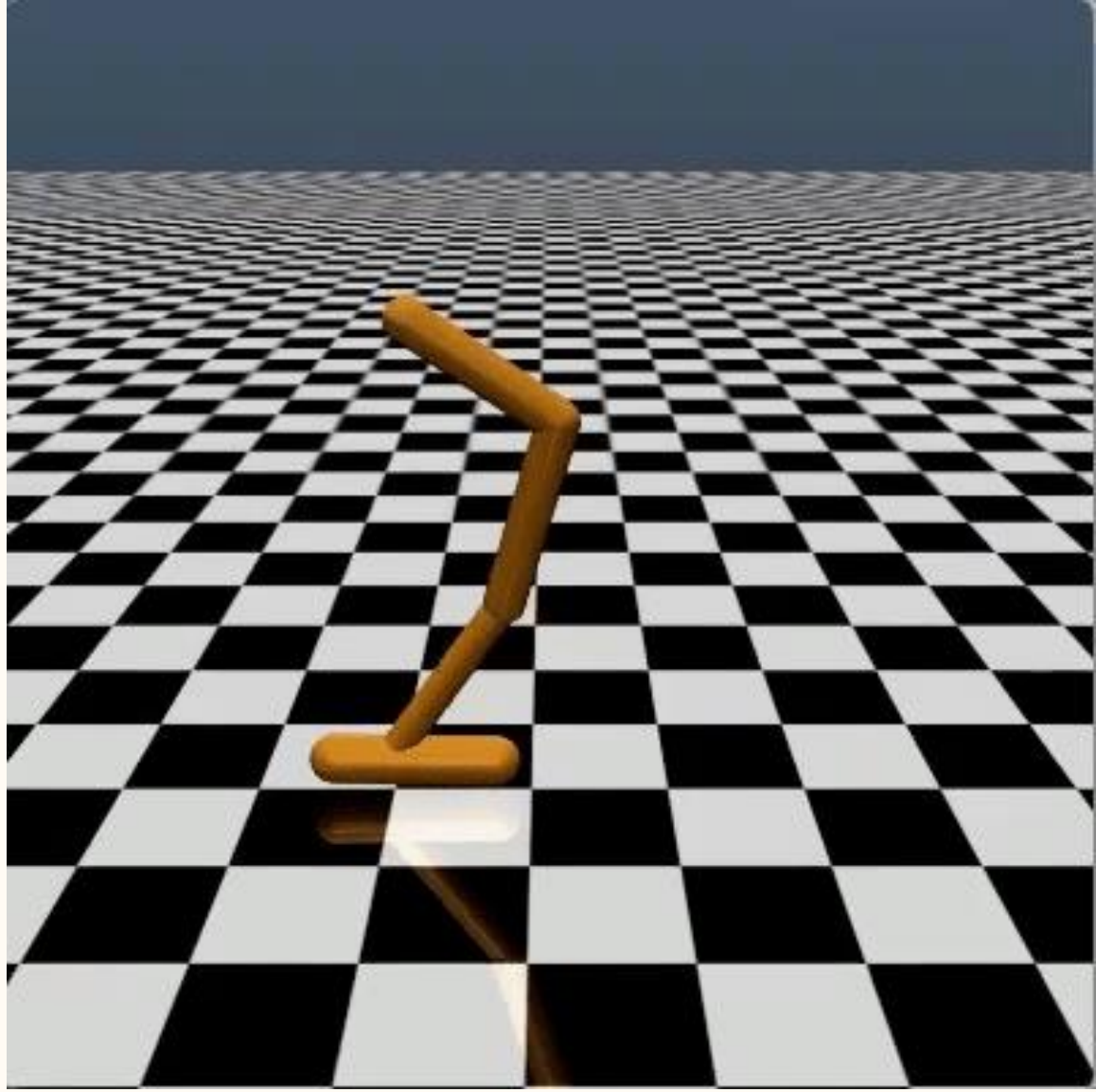
- Large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. Next token prediction does provide an effective way to steer a model away from bad outputs.
    - The pre-training dataset (must) contain some data that is unkind, so a model trained with next-token-prediction may sometimes be unkind to the user. How do we explicitly tell the model to be kind to the user?
- Next-token-prediction is not appropriate for specifying abstract goals.
    - E.g. “Follow the user’s instructions helpfully and safely.”
    - E.g. “Refuse a user’s command if it is unethical or dangerous.”

RLHF from RL

Reinforcement learning (RL), aims to control an agent to achieve high “reward”, but this reward is sometimes difficult to specify as a formal function.

Example) We know a backflip when we see it, but it is difficult program a function that returns positive reward upon a successful backflip.

RL with human feedback (RLHF) uses human feedback to determine the desired behavior, often my training a *reward model*.



See:
https://openai.com/index/learning-from-human-preferences/

P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, Deep reinforcement learning from human preferences, *NeurIPS*, 2017.
B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, Reward learning from human preferences and demonstrations in Atari, *NeurIPS*, 2018.

Aligning LLMs with RLHF

In the InstructGPT[#] paper, RLHF is carried out with three neural networks.

- $\pi_\theta(u_{\ell+1}|u_1, \ldots, u_\ell)$: Instruction fine-tuned LM, 175B GPT-3.
- r_ψ: Reward model (RM), initialized from a pre-trained LM, 6B GPT-3.
- V_ϕ: Value function model, initialized to be RM. Used in PPO.

(Smaller 6B RM was used because with a 175B RM, (1) training was more unstable which made them less suitable, and (2) using a 175B RM and value function greatly increase the compute requirements of PPO.)

[#]L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, …, P. Christiano, J. Leike, and R. Lowe, Training language models to follow instructions with human feedback, NeurIPS, 2022. (*arXiv* March 2022)

Reward model: Bradley–Terry

Reward model r_ψ is trained as a Bradley–Terry model. Specifically, train by minimizing

$$\mathcal{L}(\psi) = -\mathop{\mathbb{E}}_{(x,y_{\text{win}},y_{\text{lose}})\sim D}[\log \sigma(r_\psi(x,y_{\text{win}}) - r_\psi(x,y_{\text{lose}}))] = -\mathop{\mathbb{E}}_{(x,y_{\text{win}},y_{\text{lose}})\sim D}\left[\log\left(\frac{e^{r_\psi(x,y_{\text{win}})}}{e^{r_\psi(x,y_{\text{win}})} + e^{r_\psi(x,y_{\text{lose}})}}\right)\right]$$

where human annotator prefers y_{win} over y_{lose}.

Only relies on humans to provide comparison labels. No need to provide absolute scores.

This is soft-max regression with $K = 2$ (logistic regression) on determining probability of the two events: [(x, y_1) is better] vs. [(x, y_2) is better]

R. A. Bradley and M. E. Terry, Rank analysis of incomplete block designs: I. The method of paired comparisons, *Biometrika*, 1952.

Best-of-N sampling

Best-of-N sampling is a simple algorithm to improve generation using a reward model:

1. Generate N text outputs.
2. Select the best one as determined by the reward model.

Advantage: Simple and effective[#] way to utilize a reward model trained from human feedback. Also, no need for RL training, which can be tricky.

Downside: Sampling requires N generations, so inefficient.

[#]L. Gao, J. Schulman, and J. Hilton, Scaling laws for reward model overoptimization, *ICML*, 2023.

RLHF details

Each timestep is a BPE token.

Response generation is an episide, and an episode terminates when LM generates <EOS>.

No discount used, i.e., discount factor $\gamma = 1$ is used.

Reward (by reward model) only provided at the end of the episode. Called “contextual bandit” setting.

The PPO clip ratio is set to $\varepsilon = 0.2$.

Sampling temperature is 1 for rollouts.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, …, P. Christiano, J. Leike, and R. Lowe, Training language models to follow instructions with human feedback, NeurIPS, 2022. (*arXiv* March 2022)

Proximal policy optimization (PPO)

We now denote the language model as π_θ, viewing it as an RL policy.

- $\pi_\theta(u_{\ell+1}|u_1, \ldots, u_\ell)$ is probability of "action" $u_{\ell+1}$ based on the current "state" $u_1, \ldots, u_\ell$.

Let x be a text prompt and $y = y_{1:T}$ be its completion by π_θ. Let $y_{1:t}$ the partial completion up to token t.

PPO maintains a value function model V_ϕ.

- $V_\phi(x, y_{1:t})$: Given $(x, y_{1:t})$, what is the expected reward if we continue generation with π_θ.

Advantage $\hat{A} = r_\psi(x, y_{1:T}) - V_\phi(x, y_{1:t})$: how good is the completion $y_{t+1:T}$ compared to what V_ϕ was expecting based on $y_{1:t}$?

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, *arXiv*, 2017.

Proximal policy optimization (PPO)

If $\hat{A} = r_\psi(x, y_{1:T}) - V_\phi(x, y_{1:t}) > 0$, then $y_{t+1:T}$ was a good completion. We should adjust π_θ to make this completion more likely.

If $\hat{A} = r_\psi(x, y_{1:T}) - V_\phi(x, y_{1:t}) < 0$, then $y_{t+1:T}$ was a bad completion. We should adjust π_θ to make this completion less likely.

However, we do not want to adjust π_θ from this single (batch of) trajectories for optimization reasons. Therefore, we restrict the incentive to adjust π_θ too much using a clipped objective. (Hence “proximal” policy optimization.)

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, *arXiv*, 2017.

Proximal policy optimization (PPO)

The clipped surrogate objective in PPO is

$$k(\theta; a, s) = \begin{cases} \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, 1+\varepsilon\right)\hat{A} & \text{if } \hat{A} \geq 0 \\ \max\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, 1-\varepsilon\right)\hat{A} & \text{if } \hat{A} < 0 \end{cases}$$

Interpretation: We increase/maximize $\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}\hat{A}$ only by a small factor.

This removes the incentive to move θ far away from θ_k.

The loss is equivalent to

$$k(\theta; a, s) = \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}\hat{A}, \text{clip}_{1-\varepsilon}^{1+\varepsilon}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}\right)\hat{A}\right)$$

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, *arXiv*, 2017.

PPO v.0 (susceptible to over-optimization)

while (not converged)

sample N trajectories $(x^{(i)}, y^{(i)}_{1:T^{(i)}}) \sim (p_0, \pi_{\theta_{\text{curr}}})$ for $i = 1, \ldots, N$

$\hat{A}^{(i)}_t = r_\psi(x^{(i)}, y^{(i)}_{1:T^{(i)}}) - V_\phi(x^{(i)}, y^{(i)}_{1:t})$ for $i = 1, \ldots, N$, $t = 1, \ldots, T^{(i)} - 1$

solve:

$$\underset{\theta_{\text{next}} \in \mathbb{R}^p}{\text{maximize}} \quad \sum_{i=1}^{N} \sum_{t=0}^{T^{(i)}} \min \left(\frac{\pi_{\theta_{\text{next}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})}{\pi_{\theta_{\text{curr}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})} \hat{A}^{(i)}_t, \text{clip}^{1+\varepsilon}_{1-\varepsilon} \left(\frac{\pi_{\theta_{\text{next}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})}{\pi_{\theta_{\text{curr}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})} \right) \hat{A}^{(i)}_t \right)$$

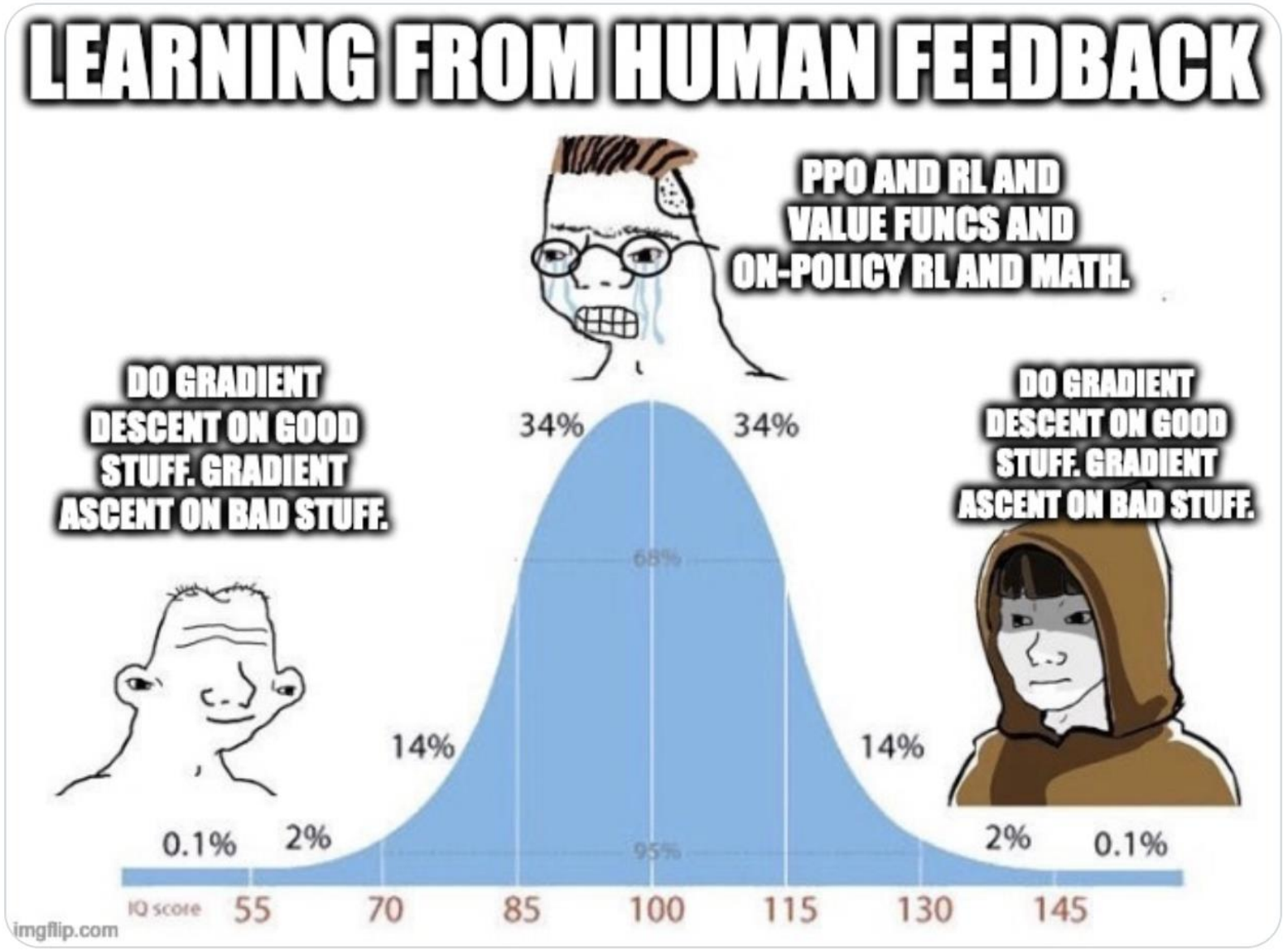
$\theta_{\text{curr}} = \theta_{\text{next}}$

solve:

$$\underset{\phi \in \mathbb{R}^q}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T^{(i)}} \sum_{t=0}^{T^{(i)}-1} \frac{1}{2} \left(r_\psi(x^{(i)}, y^{(i)}_{1:T^{(i)}}) - V_\phi(x^{(i)}, y^{(i)}_{1:t}) \right)^2$$

end

Tom Goldstein
@tomgoldsteincs



6:08 PM · Nov 29, 2023 · **13K** Views

Over-optimization

Goodhart's law: When a measure becomes a target, it ceases to be a good measure.

Problem with PPO v.0: Over-optimization.

- Reward model is not perfect, so we do not want to overfit to it. (Maximizing reward model too much will result in adversarial generation that seems good to the reward model by exploiting the imperfections of the model.)
- Moving away from the pre-trained and instruction-fine tuned model too much will cause the language model to lose its main capabilities.

Over-optimizations: Clarification

PPO's clipped objective prevents over-optimization with respect to the batch trajectories $(x^{(i)}, y^{(i)}_{1:T^{(i)}}) \sim (p_0, \pi_{\theta_{\text{curr}}})$. We do not want to move θ_{next} too far away from θ_{curr} since $(x^{(i)}, y^{(i)}_{1:T^{(i)}}) \sim (p_0, \pi_{\theta_{\text{curr}}})$ is informative about which θ_{next} is good only when θ_{next} is in a neighborhood of θ_{curr}.

Problem with PPO v.0 is that the reward model r_ψ was trained on π^{SFT}, the supervised-fine-tuned (pre-trained and instruction fine-tuned) baseline. So r_ψ is informative about π_θ^{RL} only when π_θ^{RL} is in the neighborhood of π^{SFT}. (π_θ^{RL} is initialized to π^{SFT}.)

Resolution) Impose a KL-divergence pelanty term, preventing π_θ^{RL} from moving too far away from π^{SFT}.

KL-penalty and pre-training loss

RLHF minimizes the objective:

$$\mathcal{L}(\theta) = -\mathop{\mathbb{E}}_{(x,y)\sim D_{\pi_\theta^{\mathrm{RL}}}}\left[r_\psi(x,y) - \beta\log\left(\pi_\theta^{\mathrm{RL}}(y\,|\,x)/\pi^{\mathrm{SFT}}(y\,|\,x)\right)\right] - \eta\mathop{\mathbb{E}}_{(x,y)\sim D_{\mathrm{pretrain}}}\left[\log\left(\pi_\theta^{\mathrm{RL}}(x)\right)\right]$$

The η-term means continue to train π_θ^{RL} with the pre-training loss, while we do PPO.

The $r_\psi(x,y)$-term is the RLHF with PPO. The β-term is incorporated into PPO.

PPO and pre-training update performed simultaneously or in alternating fashion.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, …, P. Christiano, J. Leike, and R. Lowe, Training language models to follow instructions with human feedback, NeurIPS, 2022. (*arXiv* March 2022)

PPO with KL penalty

while (not converged)

sample N trajectories $(x^{(i)}, y^{(i)}_{1:T^{(i)}}) \sim (p_0, \pi_{\theta_{\text{curr}}})$ for $i = 1, \ldots, N$

$$\hat{A}^{(i)}_t = r_\psi(x^{(i)}, y^{(i)}_{1:T^{(i)}}) - \beta \sum_{s=t}^{T^{(i)}} \log \left(\frac{\pi^{\text{RL}}_\theta(y^{(i)}_{s+1} \mid x^{(i)}, y^{(i)}_{1:s})}{\pi^{\text{SFT}}(y^{(i)}_{s+1} \mid x^{(i)}, y^{(i)}_{1:s})} \right) - V_\phi(x^{(i)}, y^{(i)}_{1:t}) \quad \text{for } \substack{i=1,\ldots,N \\ t=1,\ldots,T^{(i)}-1}$$

solve:

$$\underset{\theta_{\text{next}} \in \mathbb{R}^p}{\text{maximize}} \quad \sum_{i=1}^{N} \sum_{t=0}^{T^{(i)}} \min \left(\frac{\pi_{\theta_{\text{next}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})}{\pi_{\theta_{\text{curr}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})} \hat{A}^{(i)}_t, \text{clip}^{1+\varepsilon}_{1-\varepsilon} \left(\frac{\pi_{\theta_{\text{next}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})}{\pi_{\theta_{\text{curr}}}(y^{(i)}_{t+1} \mid x^{(i)}, y^{(i)}_{1:t})} \right) \hat{A}^{(i)}_t \right)$$

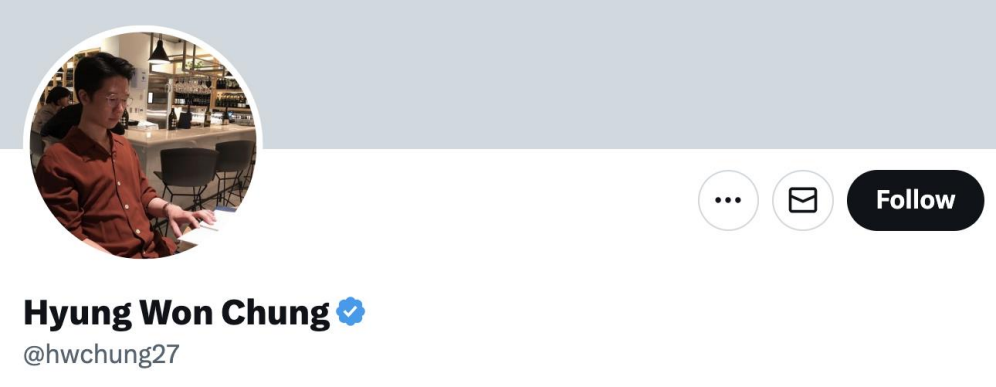
$\theta_{\text{curr}} = \theta_{\text{next}}$

solve:

$$\underset{\phi \in \mathbb{R}^q}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T^{(i)}} \sum_{t=0}^{T^{(i)}-1} \frac{1}{2} \left(r_\psi(x^{(i)}, y^{(i)}_{1:T^{(i)}}) - V_\phi(x^{(i)}, y^{(i)}_{1:t}) \right)^2$$

end

Bitter Lesson II



A counterintuitive implication of scale: *trying to solve* a more general version of the problem is an easier way to solve the original problem than directly tackling it. Attempting a more general problem encourages you to come up with a more general and simpler approach. This often leads to a more scalable method. By leveraging increasingly cheaper compute, you solve the specific problem as a by-product of tackling a more general one. Some examples: - Directly solving NLU tasks (e.g. question answering) vs. learning a general language model and solving the task as a next token prediction. - Instead of directly working on machine translation, work on a general problem of learning all languages (mT5 vs. translation-specific models). Taking this idea to the limit, it might be the case that aiming for super-intelligence is an easier (but extremely difficult) way to get to general intelligence compared to directly solving general intelligence by mimicking human intelligence. This requires thinking about approaches that are native to machines as opposed to trying to teach machines how we think humans think because if we simply mimic human intelligence, it likely won't lead to super-human intelligence.

https://twitter.com/hwchung27/status/1712209280529727705