Chat-enabled LLMs
InstructGPT and Llama 2

Venelin Kovatchev

Lecturer in Computer Science

v.o.kovatchev@bham.ac.uk

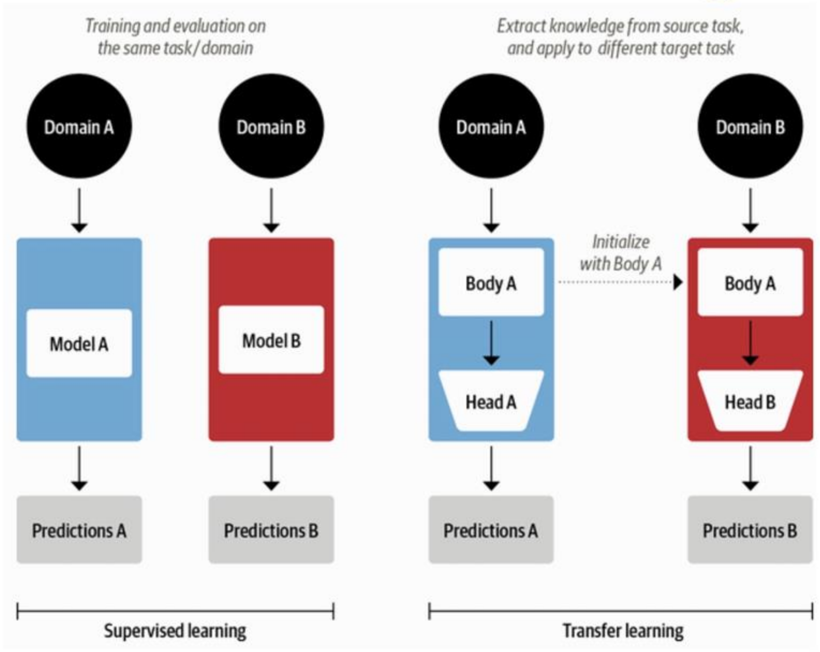
Outline

- Quick recap
  - In-context learning (part 2)
- Chat-enabled LLMs
  - InstructGPT
  - LLama2

Quick recap:
Supervised Learning, Transfer Learning
In-context learning

Supervised learning vs Transfer learning

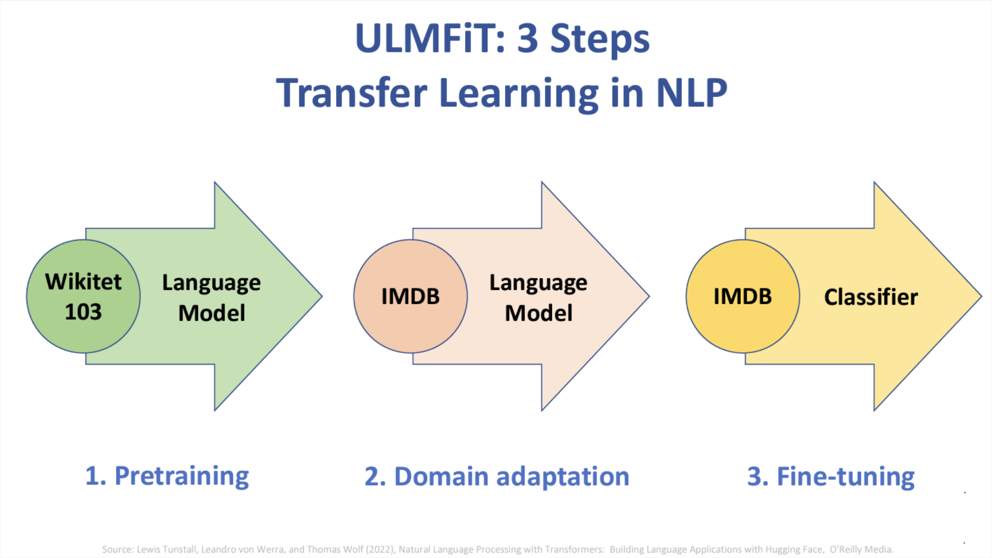
- Supervised learning
  - Full training of the model
- Transfer learning
  - Reusing part of the model
  - Finetune the head



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

ULMFiT (Howard and Ruder et al., 2018)

- "Universal Language Model Finetuning"
- Training a language model for "inductive transfer learning"
  - Model trained on a source task (language modeling)
  - Finetuned with limited data on target task
- Language modeling – the analogy of ImageNet
- Base model - BiLSTM



ULMFiT Pipeline

- Three step training
- Most parameters remain intact

They're only changing the bottom part.

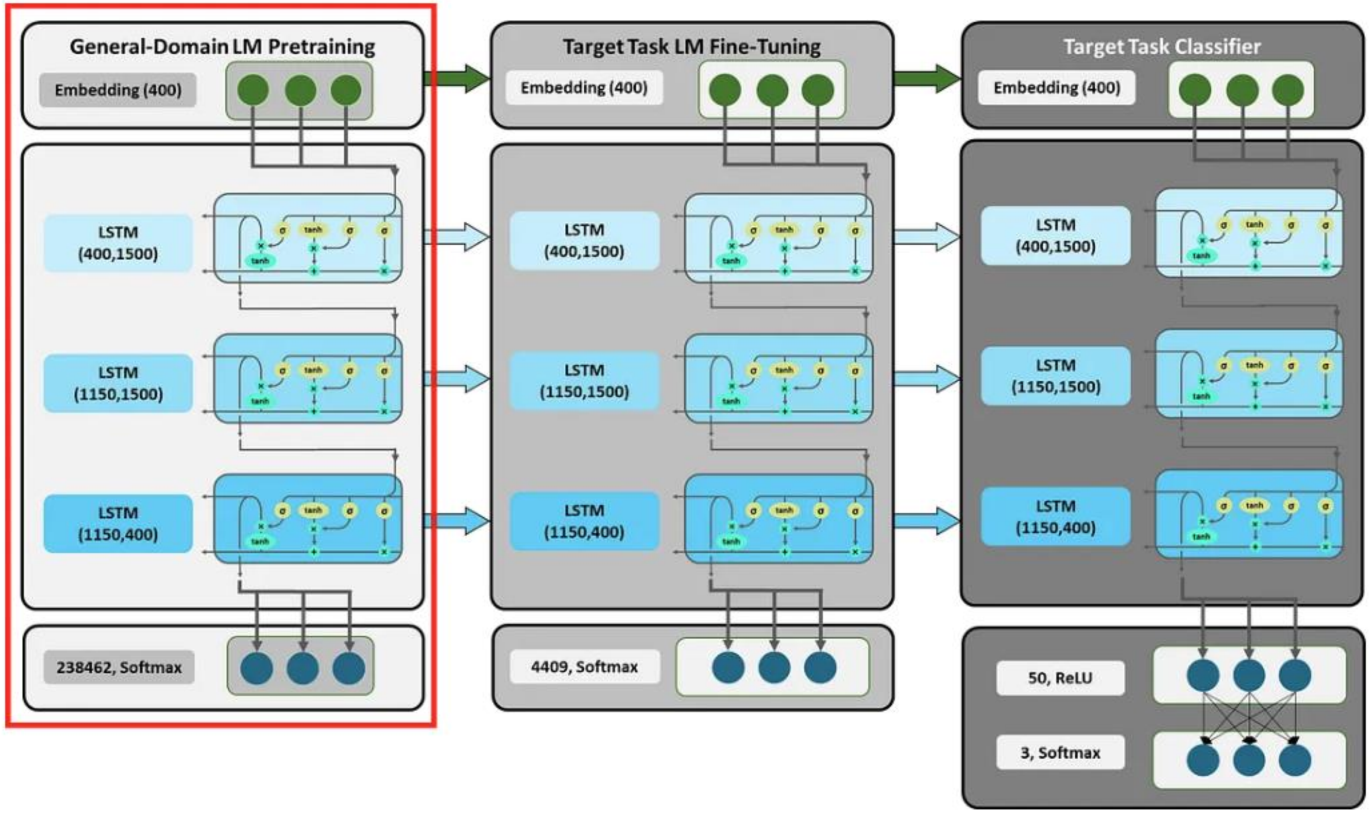


Figure 5: Block Diagram of ULMFiT in Text Analysis — Image from by HU-Berlin from GitHub

Transfer learning and Transformers

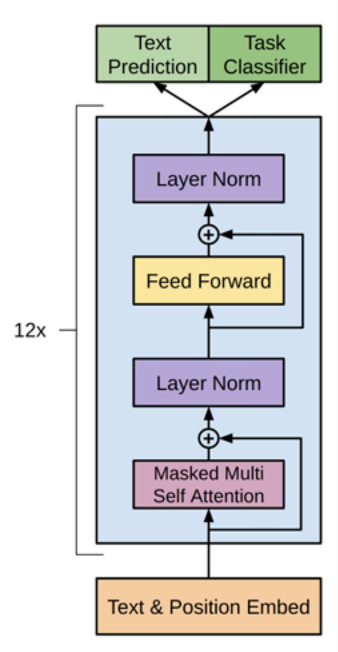
- Transformers quickly adopted the idea of transfer learning

- Three types of transformers
  - Encoder models: BERT, ROBERTA, DistilBERT
  - Decoder models: GPT, GPT2
  - Encoder-decoder models: BART, T5

- Pop-quiz: what is the difference between the three types of transformers?

The decoder transformer: GPT

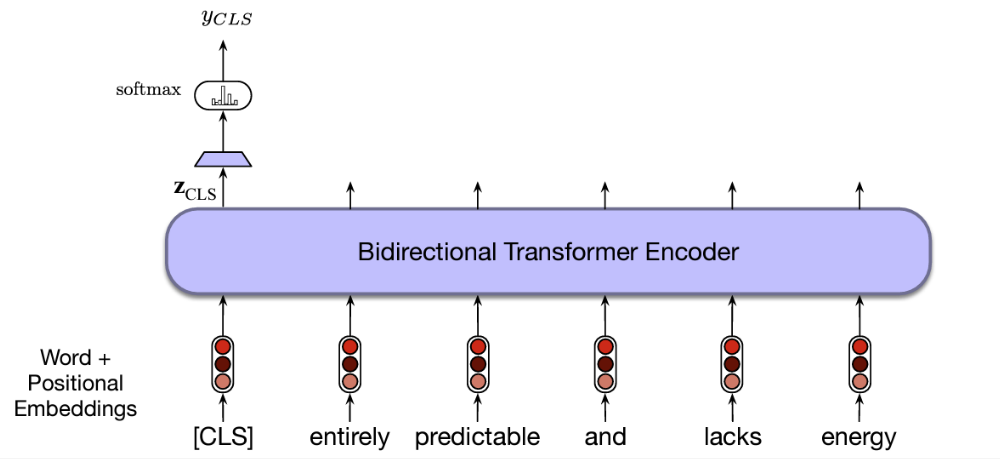
- GPT1 is a decoder (a.k.a. autoregressive) transformer
  - Causal attention (!) + a standard transformer block
  - Trained on neural language modeling
  - Transfer learning capabilities
- Intuition:
  - Generative pre-training
  - Discriminative finetuning
  - Two separate loss functions $L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$

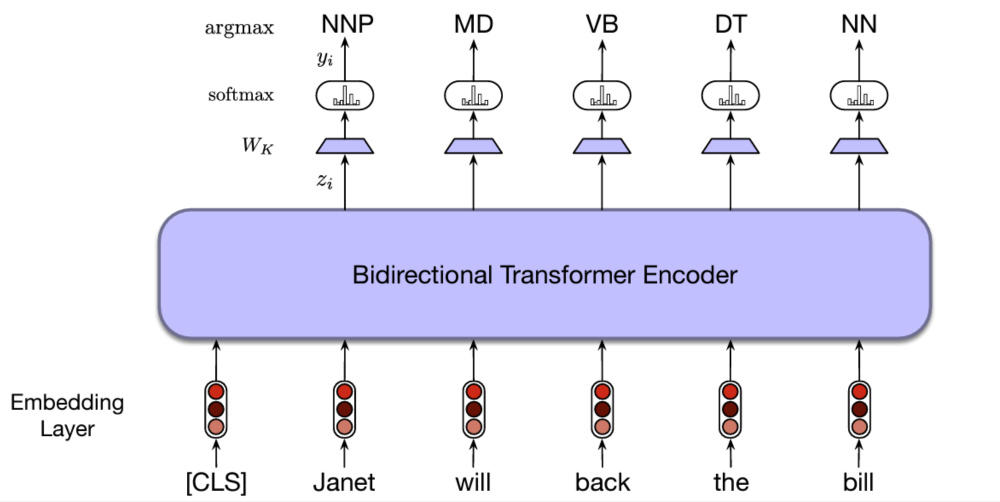


The encoder transformer: BERT

- BERT is the original encoder-only transformer
  - Bidirectional attention + a standard transformer block
  - Trained on two tasks:
    - Masked Language Modeling
    - Next Sentence Prediction
    - Pop quiz: what did each of those do?
  - Can be finetuned on a variety of tasks
    - Generally, performs better than GPT

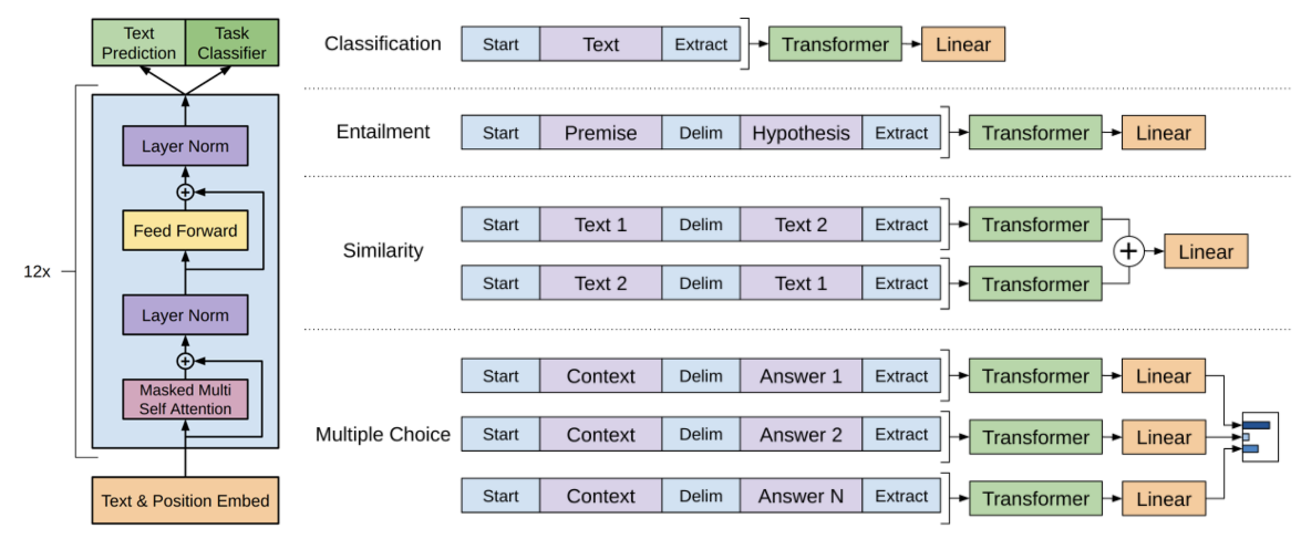
Instead of predicting the next word, predicting the current word.





Task specific input transformations

- Task reformulating
- Using special tokens (sep, start/end)
- Comparing separate "streams"
- Task design is a non-trivial task
  - Task formulation; Data format; Metrics and Evaluation



GPT3

- A major shift in NLP paradigm
  - Arguably the largest shift since moving from pipeline to end-to-end models
- Introduced few-shot and zero-shot learning
  - Teaching a model to perform a task without changing the weights (!)
- In-context learning and prompt engineering

Few-shot and Zero-shot learning

- The problem
  - Getting training data is complex and expensive (there are far more tasks than datasets)
  - Overfitting (to spurious correlations)
  - Humans don’t need large training data for all tasks
- The goal
  - One model that can perform multiple tasks
  - In-context learning
  - AGI?

In-context learning vs supervised/transfer learning

- Using the input to specify the task
- Consider the following inputs to a transformer model:
  - "I like this movie, it's the best in the Avengers series!"
  - "I bike to work every day. <SEP> I drive to work every day."
- What is the task? What is the output?
  - The task is what you train the model to do
  - The first sentence can be an input to a NER model
  - The second sentence can be an NLI task or a similarity task

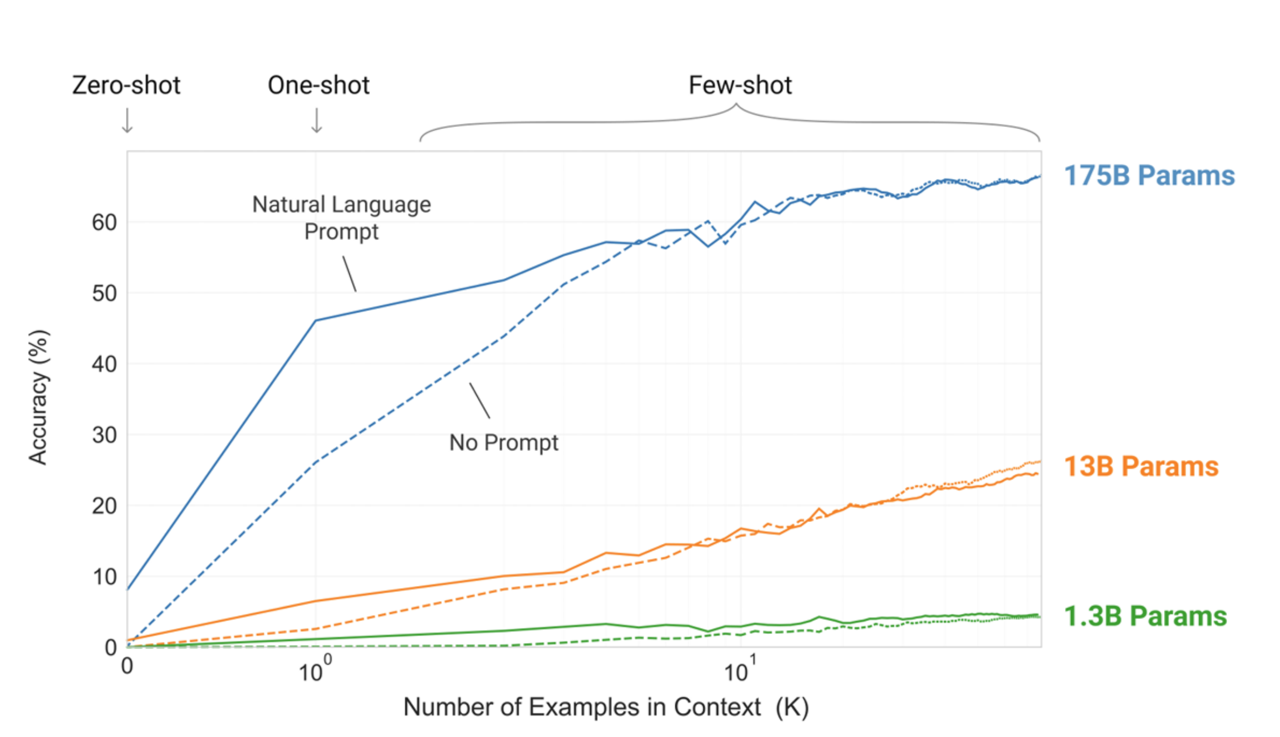
In context learning

giving a specific prefix for the text.

- Now consider the following inputs:
  - “What is the sentiment of the following text: I like this movie, it’s the best in the Avengers series!”
  - “Do those sentences contradict each other: I bike to work every day. <SEP> I drive to work every day.”
- Is anything missing in that formulation?
- How do we achieve in-context learning?
  - GPT3 paper argues that scale and emerging properties are the answer
  - Increasing model size from 17B to 175B

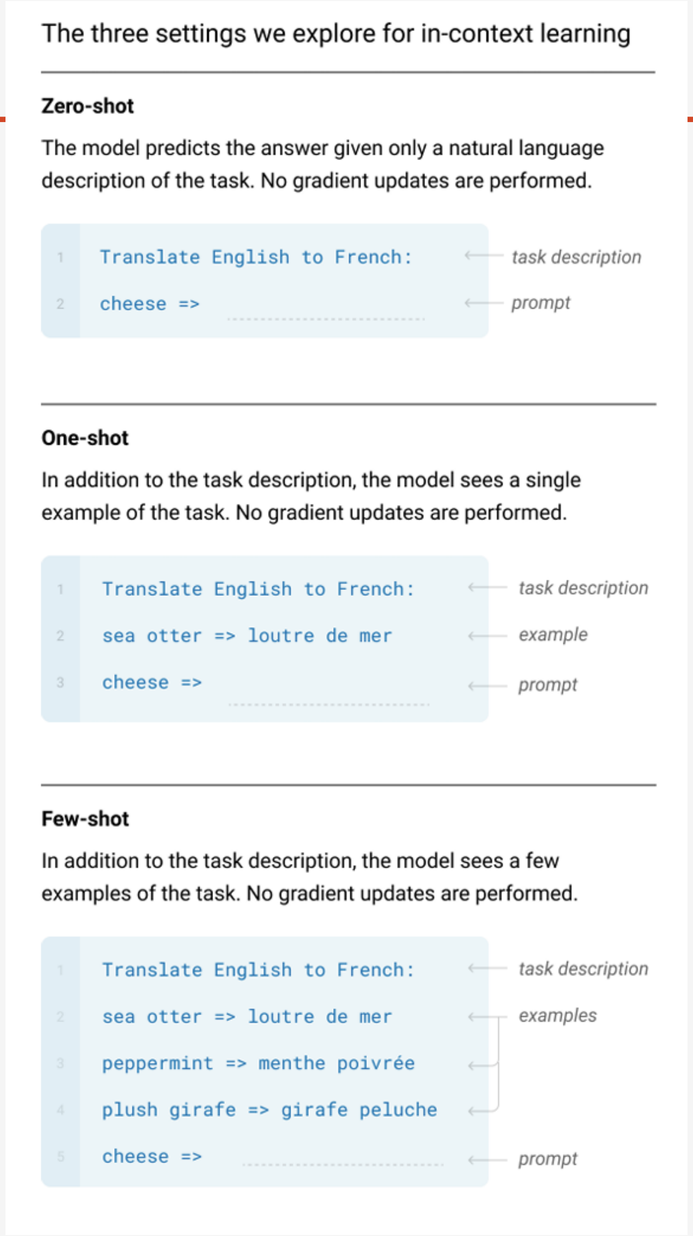
In-context learning and model size

- Two key factors
  - Model size
  - Number of examples
- Model size -> “emerging properties”



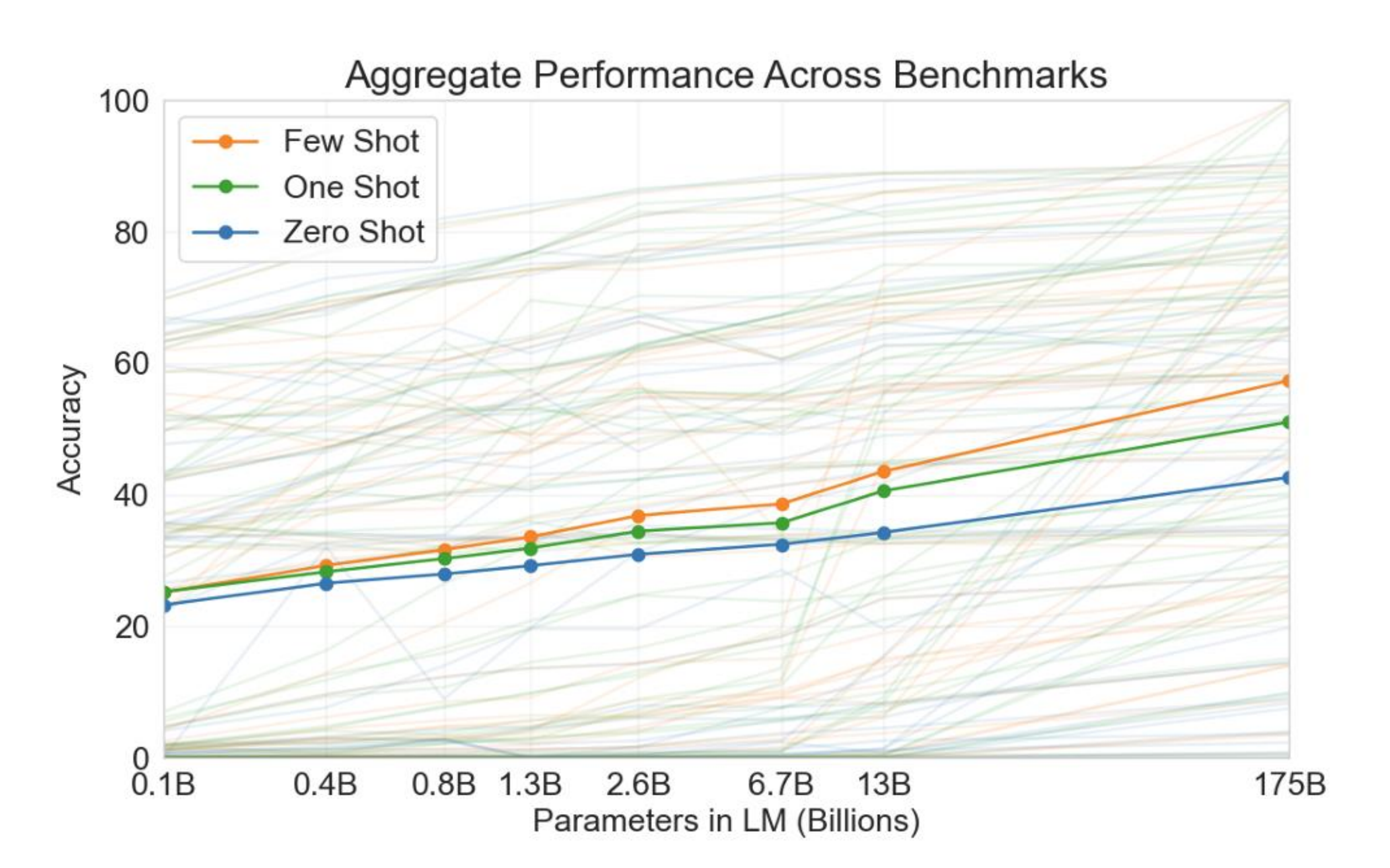
Zero- One- and Few-shot learning

- Three different experimental conditions
- No gradient update or finetuning
- The only difference – number of examples



In-context learning and model size

- Same model
- Different sizes
- Different number of examples

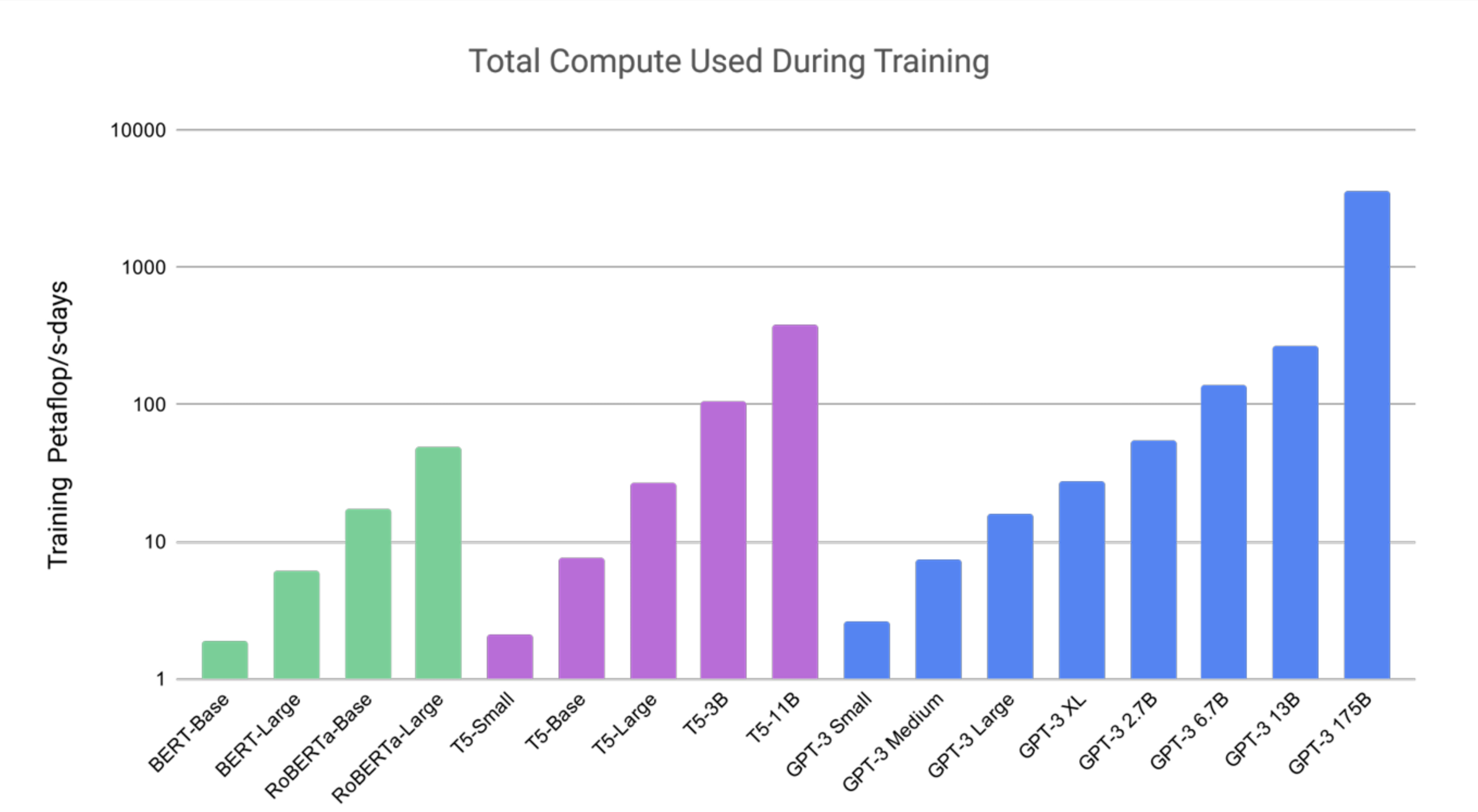


Mode architecture

- Same as GPT2, scaled to 175B params
  - 96 layers
  - 96 attention heads
  - 128 d per head -> 12k d for the hidden state

- Causal attention, trained on LM task
  - Trained on Common Crawl (1 trillion words) + data filtering + they did a data curation
  - Additional curated high-quality datasets

Compute used for training



GPT3 and current LLMs

- Almost all of current chat-enabled LLMs are based off the concepts in GPT3
- Newer models are performing better
    - More parameters; larger and better datasets
    - Additional training: supervised and RLHF finetuning
    - Human-driven and machine-driven prompt engineering
- GPT3 can, in principle, do anything that modern LLMs can
    - The difference is in the implementation, not in the core concepts (!)

Chat-enabled LLMs

Bringing LLMs to end users

- GPT3 opens the possibility of multi-purpose LLMs
  - No need for finetuning or updating parameters
- The Turing test and NLP/AI goal of conversations and collaborations
- InstructGPT and ChatGPT
  - Improving GPT ability to converse
  - Improved experience for end-users

InstructGPT – training models to follow instructions

- Scale is not everything
  - Hallucinations AIが もっともらしいうそを つくこと
  - Toxicity
  - Lack of helpfulness
- Improve the training (and evaluation) procedures
- "Align" models with their users

InstructGPT – high level idea

- Determining human preferences and goals
  - Preference of responses given a label
  - Usefulness and Safety

- “Align” the model output with human preferences
  - Additional model training
  - Focused improvements

The LM training objective

- Language modeling is not “following instructions”
- Language modeling does not take (individual) preferences
- Every training sequence is equally important
- Preference in output (in language modeling) depends on
  - The observed frequency in training data
  - The sampling strategy
- Few- and Zero-shot learning are an “emerging” side effect, not an intentionally defined goal

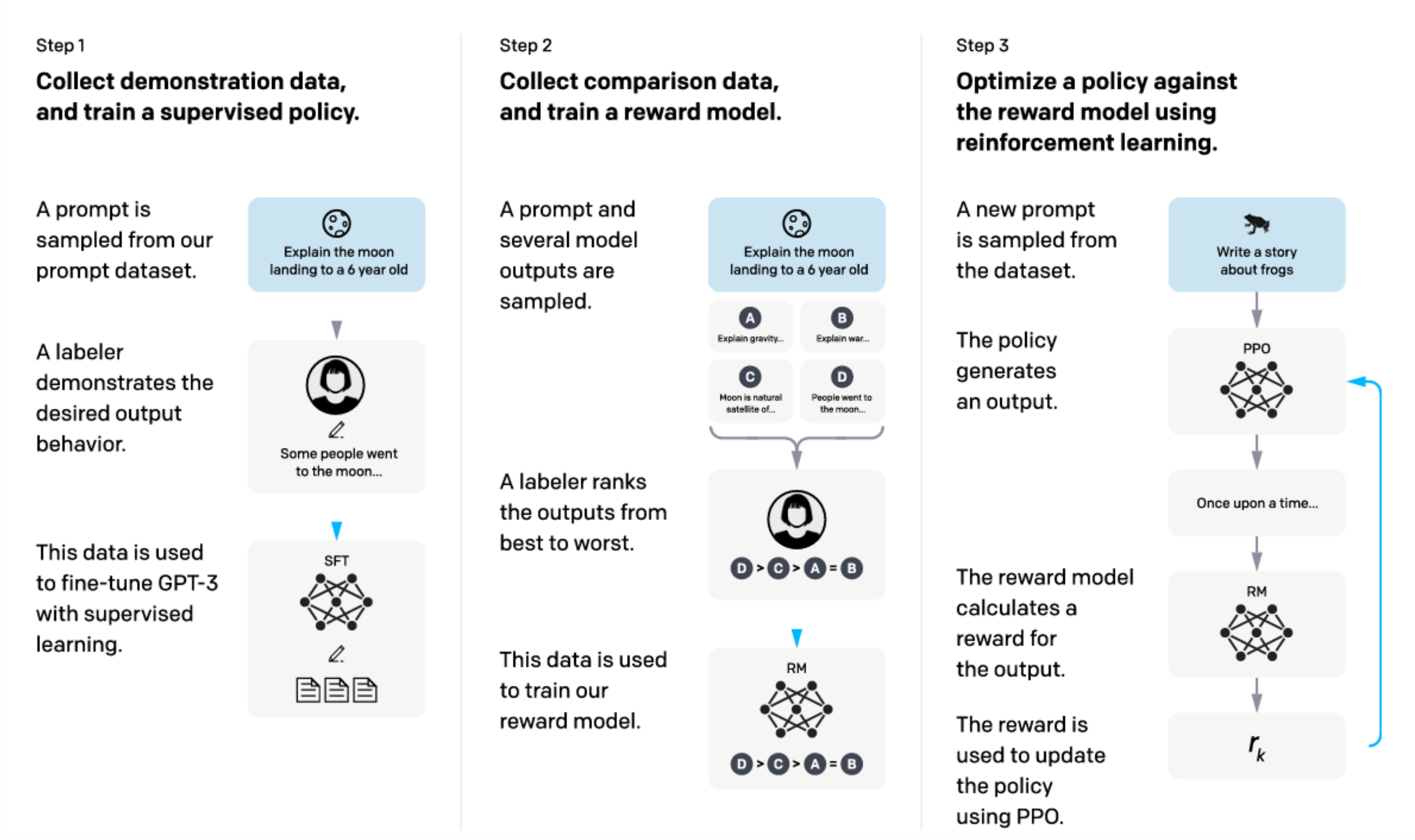
InstructGPT – the goals of “alignment”

- Three key objectives
  - Helpful
  - Honest
  - Harmless
- Later, reduced to two:
  - Helpful
  - Harmless

InstructGPT – steps in training

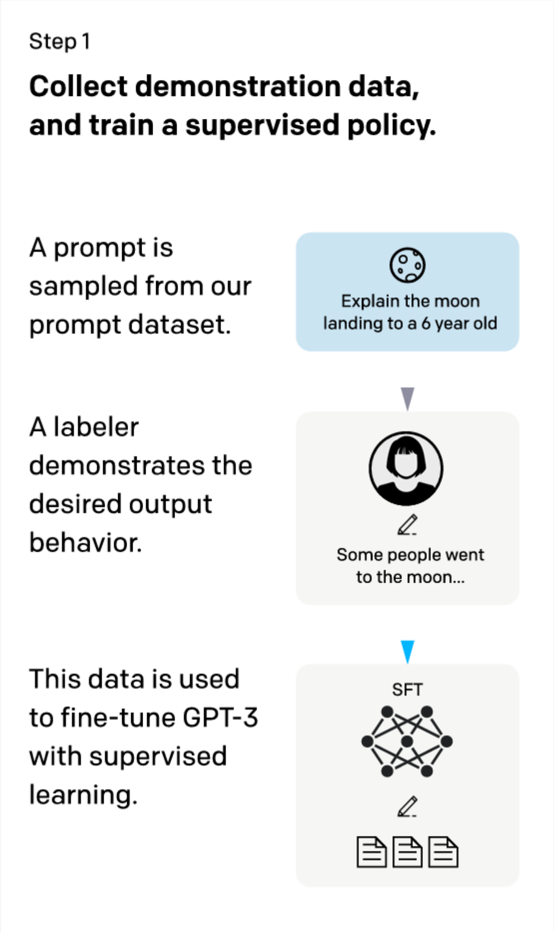
- Three (four) step training process
  - General pre-training on LM (reusing GPT3)
  - Supervised LM finetuning on human preferences
  - RL LM finetuning on human preferences
    - Intermediate step on training a reward model
- How does this compare to other techniques that we have seen before?

The training process of InstructGPT



Collecting demonstrations and supervised training

- Input data
  - Real user prompts (first version has human-written prompts)
  - Human written responses
  - Prioritizing helpfulness over truthfulness and safety for training (!)
  - Reverting the priorities for testing
- Training process
  - Language modeling objective
  - Using only training data from the "SFT" dataset



Is SFT enough to get good models?

- SFT improves the performance of LLMs for chat

- There are some (empirical) limitations
  - Availability of human labeled data
  - Overfitting to training data and diminishing returns
  - Limitations of coverage and limited improvement w.r.t bias

Using RLHF to further improve alignment

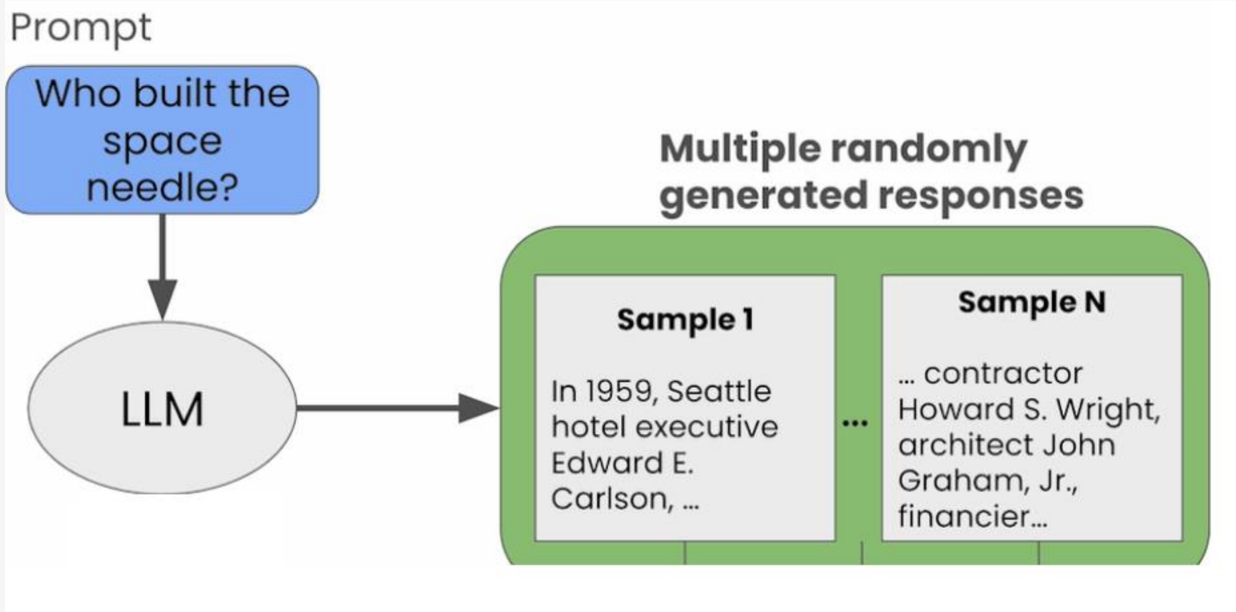
- InstructGPT proposes using RL to continue alignment after SFT

- Three step process
  - Obtaining data about preferences
  - Training a reward model
  - Aligning using RLHF

Collecting comparisons and training a reward model

- Labelers are given one prompt and multiple responses

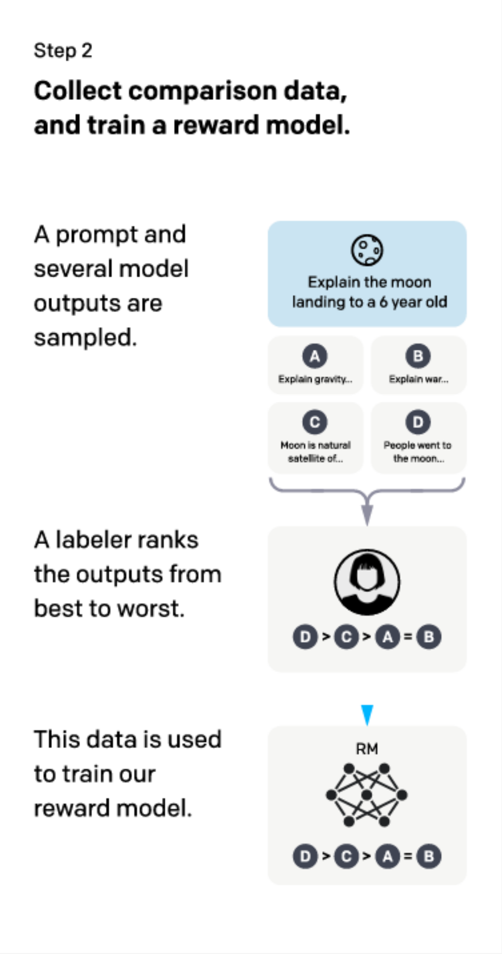
- How many responses do you show?
    - Two responses
    - Ranking of multiple responses
    - Ranking + binarization
- Multiple annotators + inter-annotator agreement



Source: Medium.com

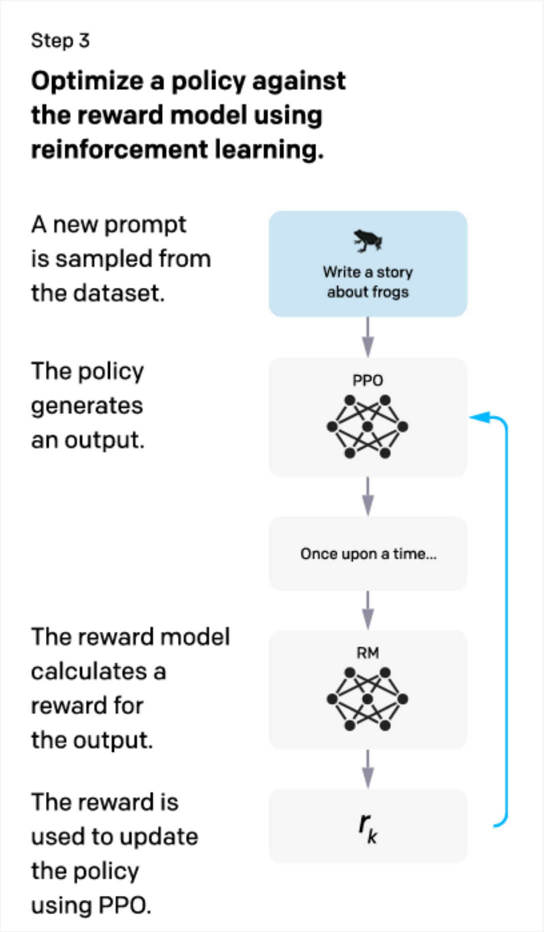
Training a reward model

- Supervised finetuning of an SFT trained LLM
    - A smaller version is sufficient (e.g. 6B params)
- Task specific head (regression):
    - Provide a scalar score for an input (prompt + response)
- Training:
    - Get the score of two (or more) comparisons
    - Calculate the diff. between pref. and non-pref. labels and convert to log prob.
    - Update all comparisons for a single prompt to avoid overfitting
    - $$\operatorname{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right]$$



Reinforcement Learning for Policy Optimization

- The final model parameters are optimized via RL
- The model generates a response to a prompt
- The reward model assigns a reward
- The policy is updated given the reward and the probability of the response



The objective function of RLHF

- The reward function takes into account
  - The reward for the prompt – r(x,y)
  - The probability of the response, given the RL policy
  - The probability of the response, given a SFT model

$$\text{objective}\,(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] +$$
$$\gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]$$