Large Language Models

Instruction tuning & model alignment

Flavio Giobergia

From next token prediction to assistants

- The LLMs discussed so far are next token predictors
- "ChatGPT"-like assistants need to:
  - Follow instructions
  - Provide helpful answers
    - And be polite, not insult us, …
- HHH objectives:
  - *Helpful*, *Honest*, *Harmless*

```
what is 2 + 2?

What is 2 plus 2? What is the
answer to 2 plus 2? What is the
answer to 2 plus 2 in math?
```

```
what is 2 + 2?

The answer to 2 + 2 is 4.
```

Outputs from Llama 2 7b
(top: pre-trained, bottom: fine-tuned)

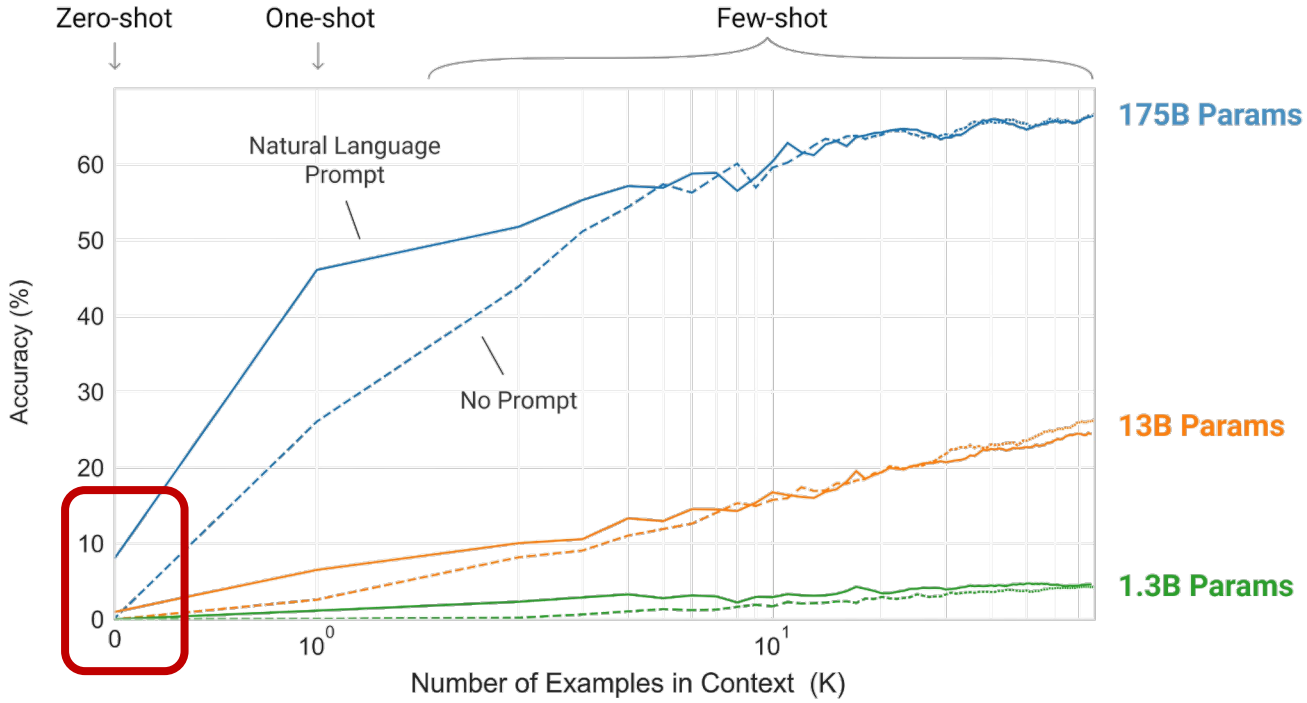
Language models are few-shot learners!

- Models have been shown to generalize to new tasks in "few-shot" mode (e.g. in Brown et al., 2020)

https://arxiv.org/abs/2005.14165

- *Zero-shot performance still low*!
  - Explaining the task without examples was not working well

- We'd like assistants to generalize without providing new examples for each task



Including instructions in prompts

- T5 provides a general architecture that includes the task to be performed as a part of the prompt.
- The model learns to condition the answer based on the request in the input context.

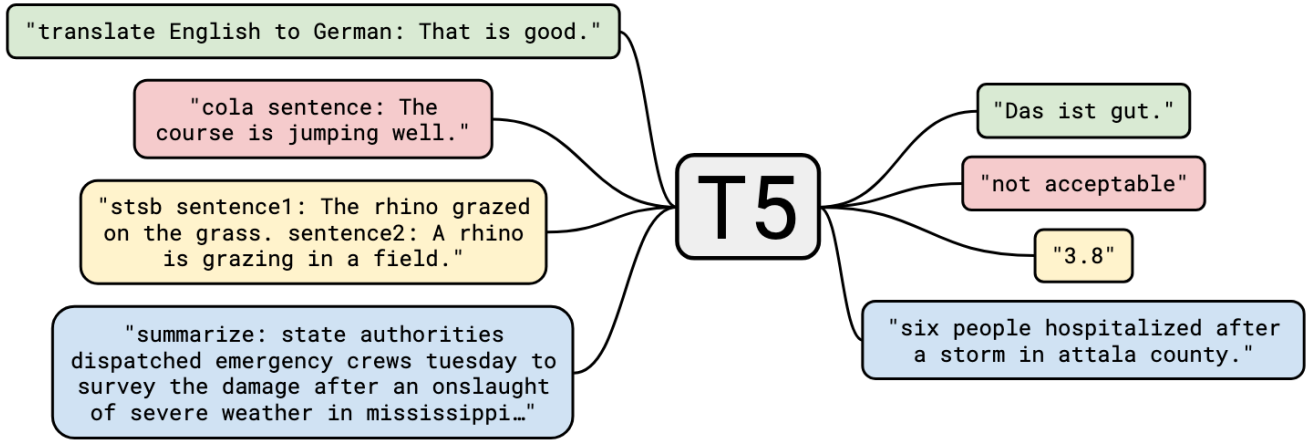


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

Limitations of T5

- T5 has limitations
  - Does not generalize to new tasks
    - It cannot “Translate german to english”
  - Expects the task to have a specific format

Input: `can you translate from English to German, What is your profession?`

Output: `Was ist Ihr Beruf?` ✅

Input: `can you translate from English to German the following sentence? What is your profession?`

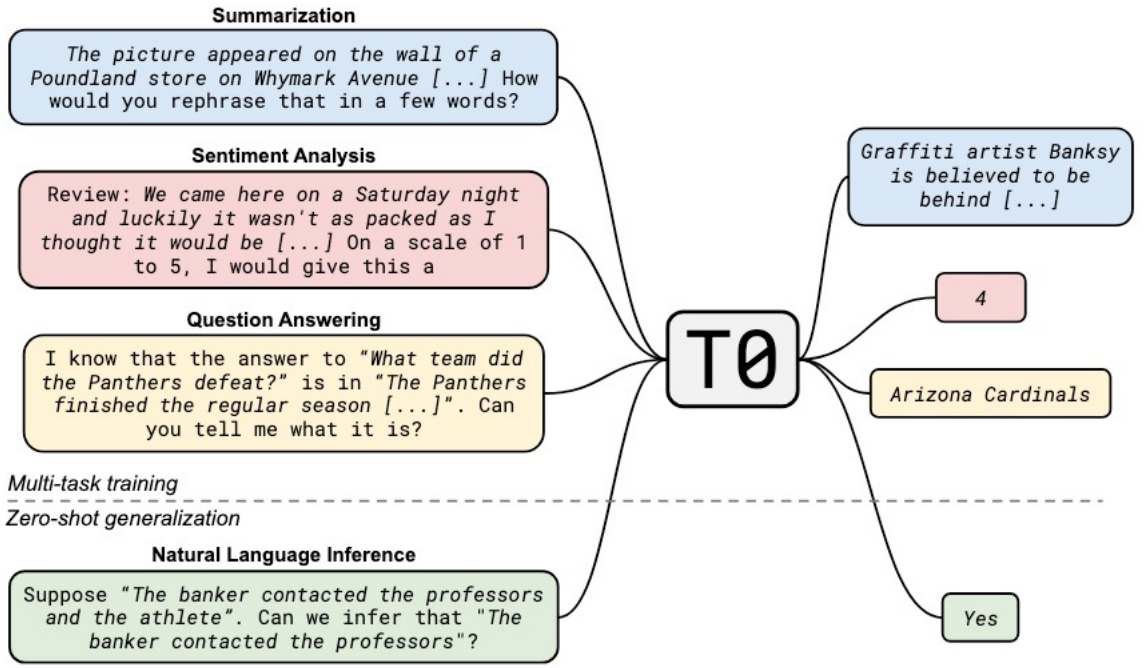
Output: `<unk> <unk>…` ❌

Input: `compute: 2+2 =`

Output: `:2+2+2+2+2+2+2+2+2+…` ❌

Generalizing to new tasks – T0

- T0 is a T5-inspired model
- *Pretrained* on masked LM task
- *Fine-tuned* on a mixture of multitask Q/A pairs
- Goals (Research Questions):
  - Can the model handle differently-worded prompts?
  - Can this fine-tuning help the model generalize to other tasks?

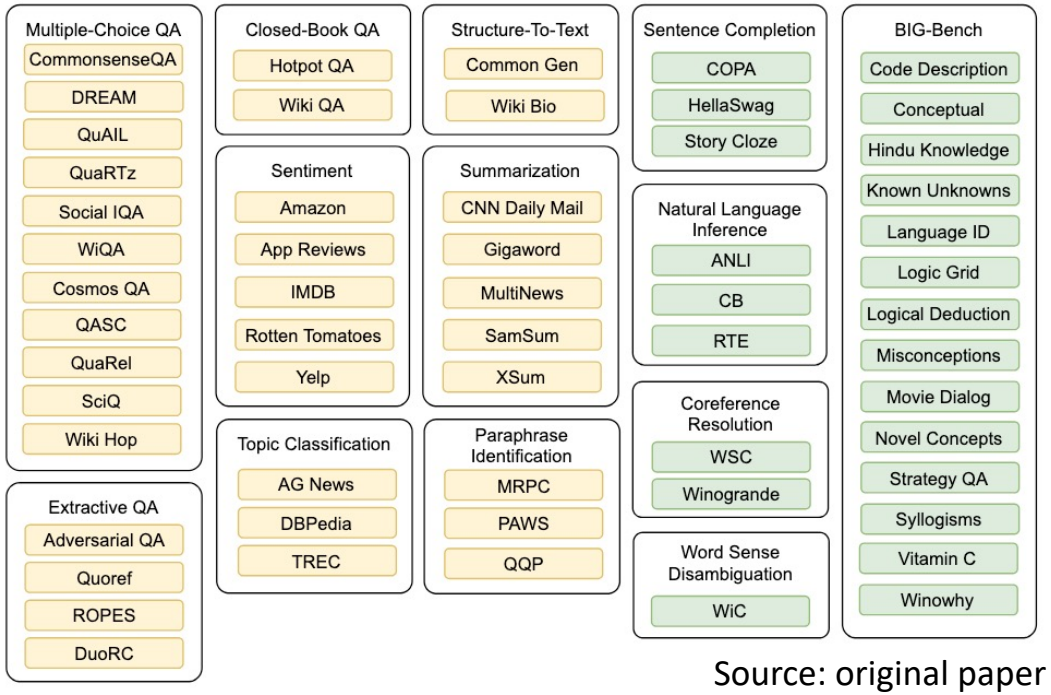


Source: original paper

Sanh, Victor, et al. "Multitask prompted training enables zero-shot task generalization." *ICLR 2022,* https://arxiv.org/pdf/2110.08207

Fine-tuning datasets

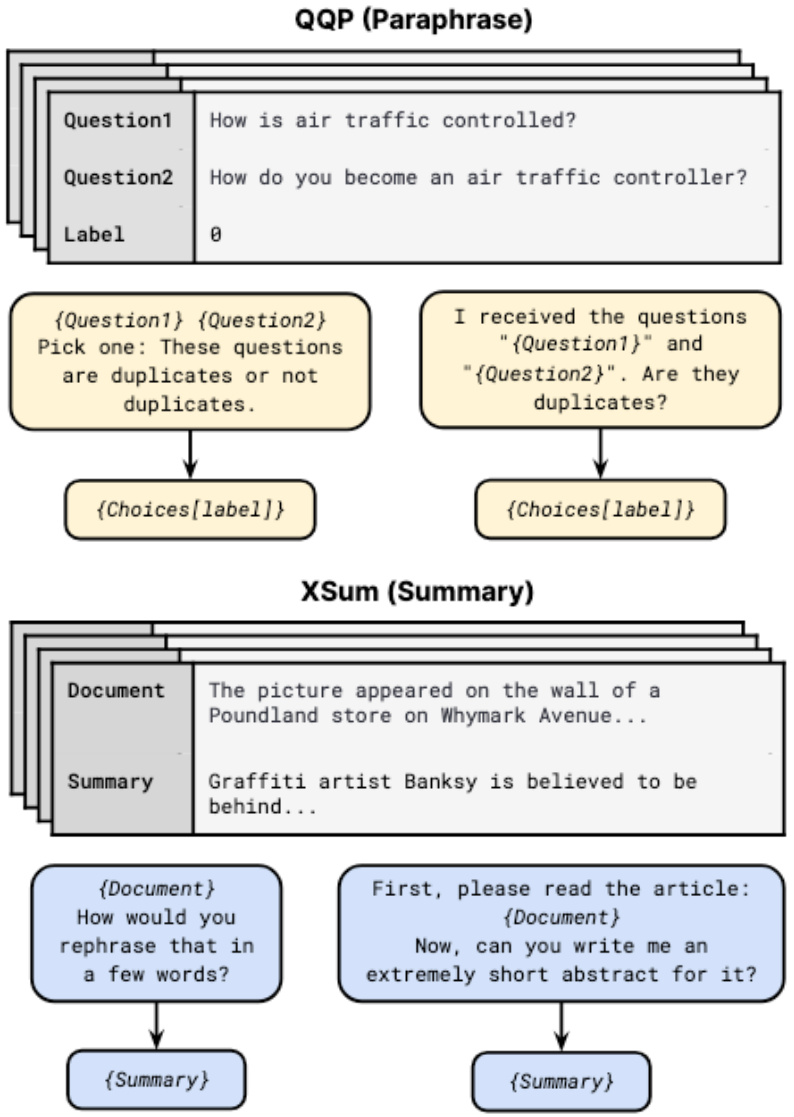
- Various datasets are identified and used for the fine-tuning
- To verify zero-shot generalization to new tasks, the datasets are divided into:
  - *Fine-tuning datasets*
  - *Evaluation datasets*
- All datasets for a task (e.g. “Natural Language Inference”) are either used for training, or for testing



Source: original paper

Templates for prompting

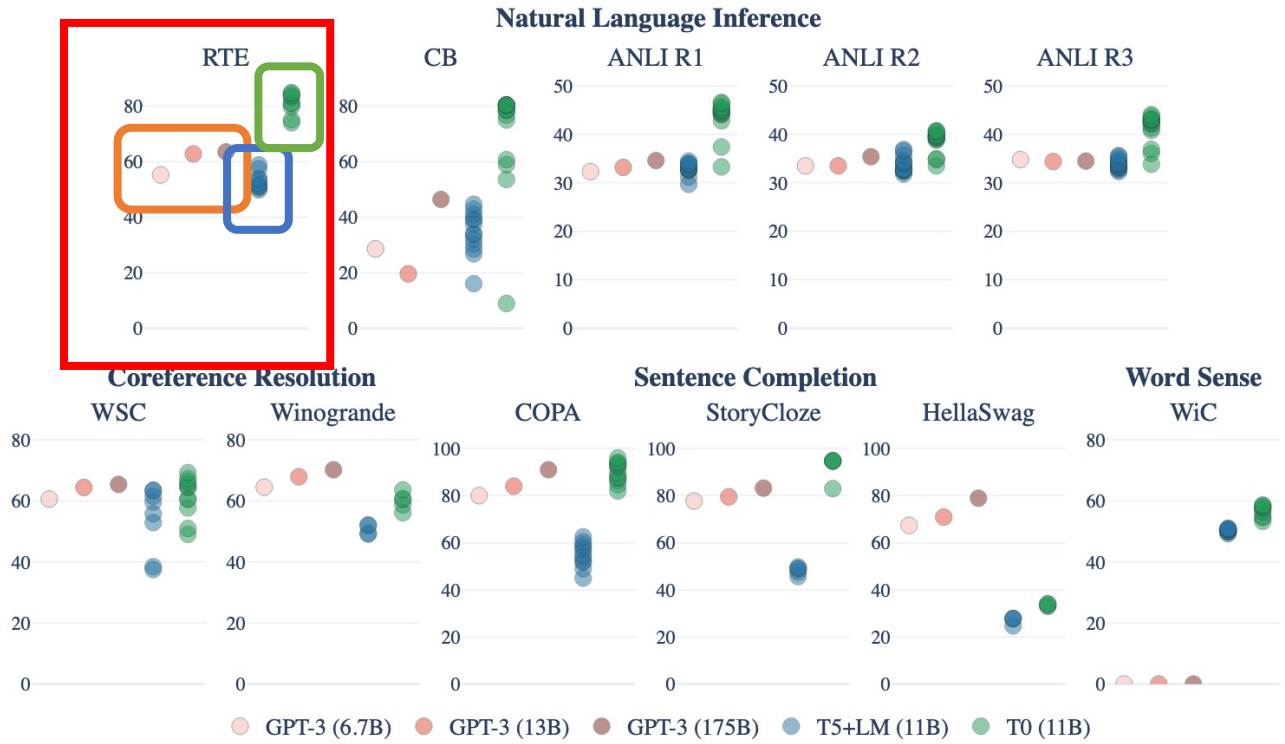
- It is generally difficult to obtain large datasets in the form of Question-Answer pairs
- To overcome the problem, the authors of T0 used *templates*
  - Each task was phrased as a question
  - Multiple rephrasings for the same task
  - Sometimes,inverting the task
    - (“What would be a good question for this answer? <answer>”)
- Ideally, the model generalizes to other forms of asking the same question
  - (Because of the semantic similarity shared)



Source: original paper

Generalization to new tasks

- Indeed, T0 shows a remarkable improvement in performance on *new tasks*, in *zero-shot*
  - E.g., NLI
- Consistently beats T5
- Generally also better than GPT-3

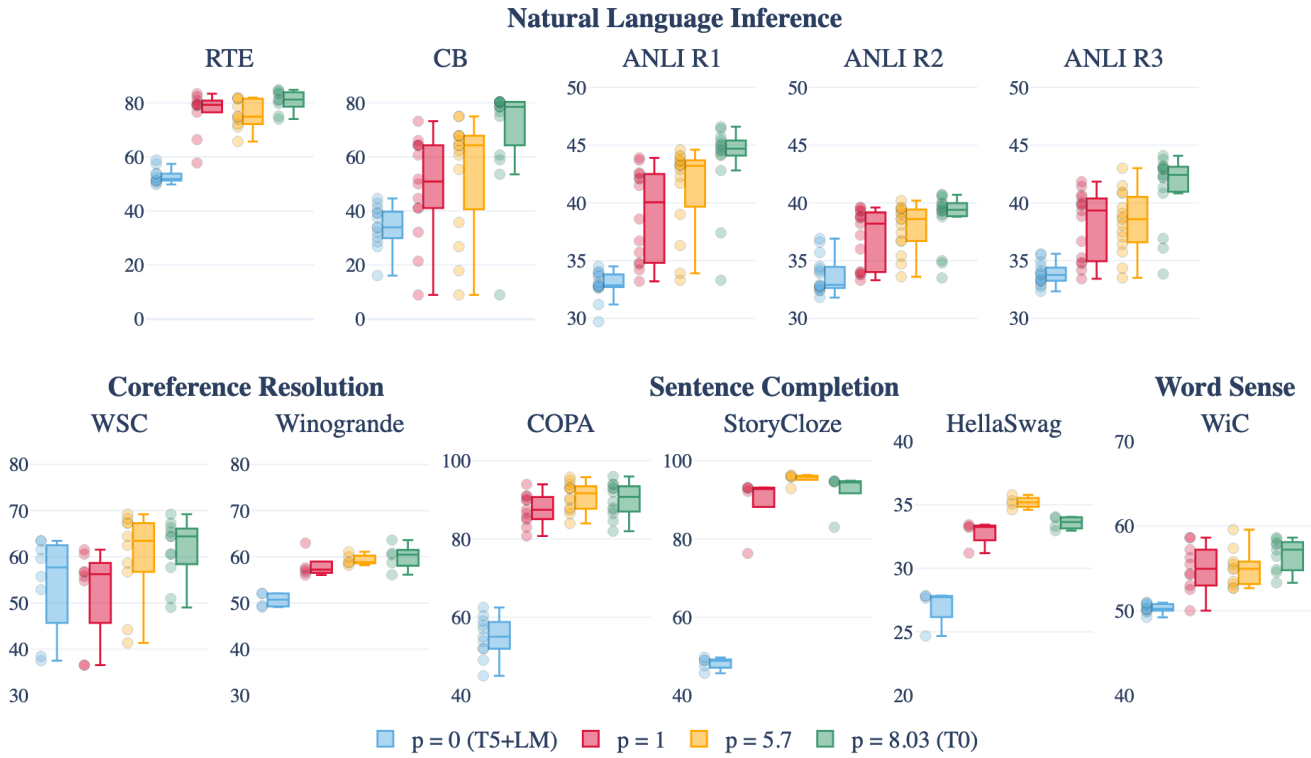


Performance on the y axis (accuracy/rank classification depending on the task).
Source: original paper

Prompt robustness

- The authors show that adding more prompt versions improves the model performance even for new tasks

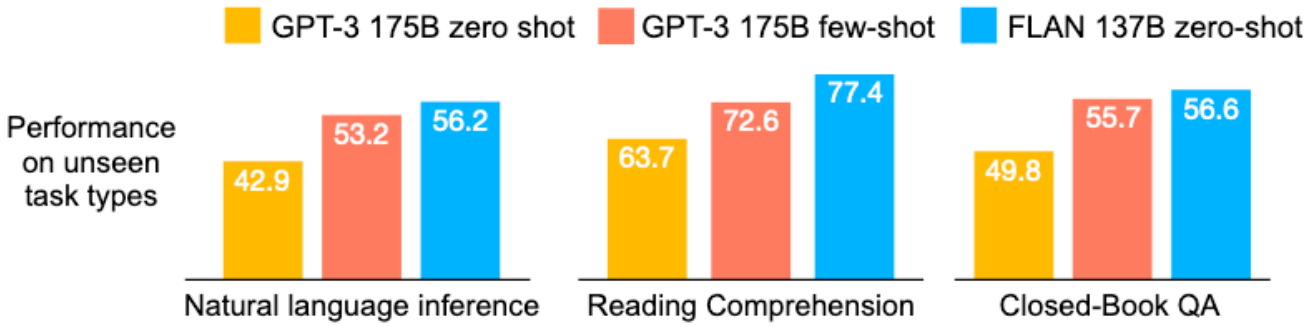
- This seems to indicate that the model is getting overall better capabilities of providing answers for new tasks by seeing more types of questions



p = average number of prompts used per dataset
Source: original paper

Finetuned Language Net (FLAN)

- *"Finetuned Language Models are Zero-Shot Learners"*
  - 2022 papers by Google Research
- Similar concept as T0
  - Both published in ICLR 2022
- Applied to larger models (up to 137B)
- Main result: *instruction tuning substantially improves zero-shot performance on unseen tasks*



Source: original paper

Wei, Jason, et al. "Finetuned language models are zero-shot learners." *ICLR 2022*, https://arxiv.org/pdf/2109.01652

FLAN setup/results

- Similar settings (held out tasks), 10 prompts for each dataset
- Comparisons against other non instruction-tuned models
- Consistent results: instruction-tuned versions generally perform better than non-tuned counterparts

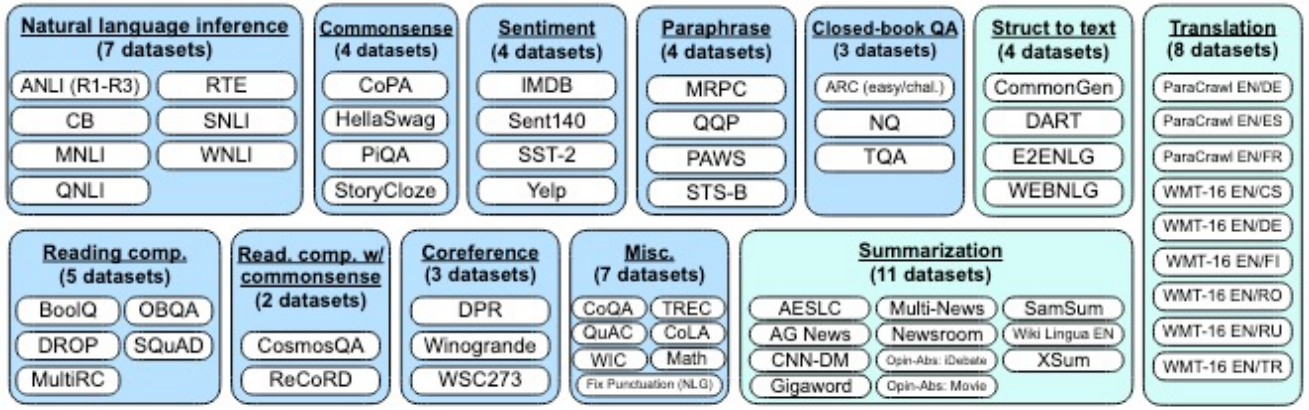


Figure 3: Datasets and task clusters used in this paper (NLU tasks in blue; NLG tasks in teal).

Source: original paper

Aligning to human preferences

- Making a model larger may improve its performance in next token prediction capabilities
  - (and zero-shot task generalization, as discussed)
- However, models can still be *untruthful*, *toxic*, *not helpful*
- In other words, models are *not aligned with human preferences*

Problems with classic training of LMs

- In *Learning to summarize with human feeback*, authors highlight that LMs are limited by:
  - *Poor metrics* (e.g., ROUGE), not capturing information about quality of outputs
  - *Poor objectives* (e.g., cross-entropy) do not distinguish between *important errors* and *minor ones*
    - E.g., making up facts and using synonyms are penalized in the same way in the loss
  - During training, models do not distinguish between *high-* and *low-quality data*
    - Models learn equally across all types of qualities

- The goal of the work is to improve alignment of LM's outputs with what humans actually think is useful

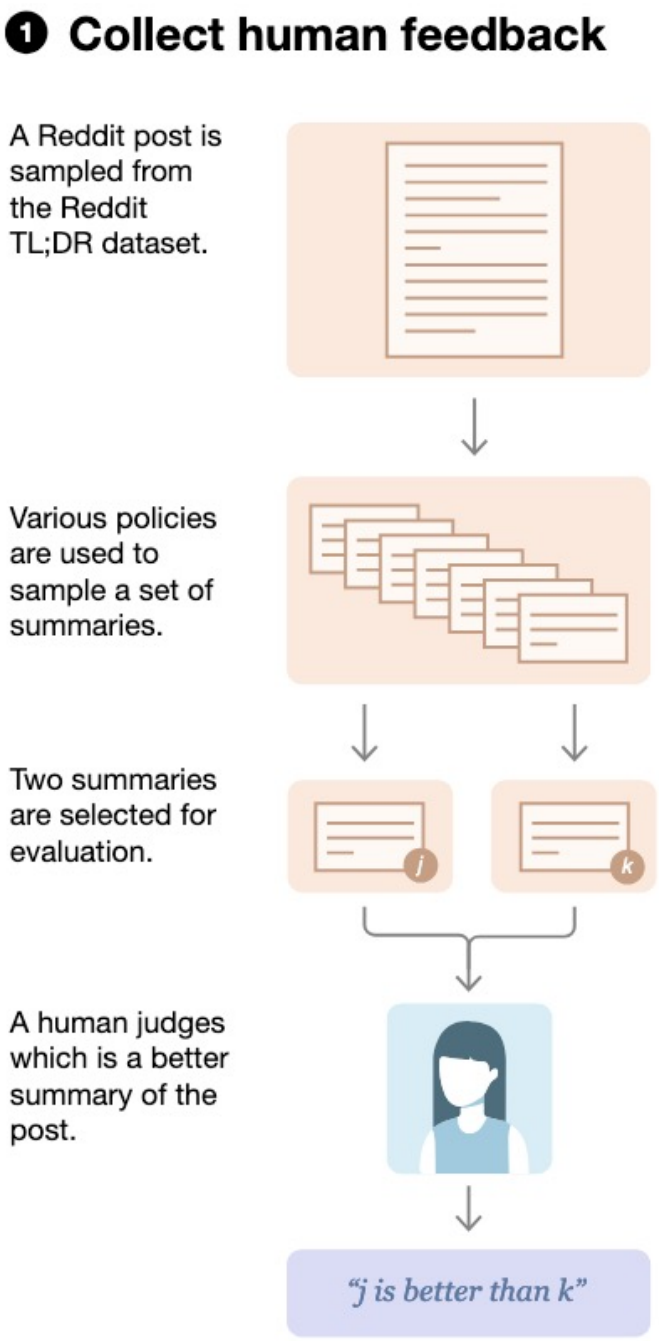
Stiennon, Nisan, et al. "Learning to summarize with human feedback." *Advances in Neural Information Processing Systems* 33 (2020): 3008-3021. https://arxiv.org/pdf/2009.01325

Three-step approach

- The proposed approach consists of three steps:
  1. *Collect human feedback*
  2. *Train reward model*
  3. *Fine-tune model to learn "human" feedback*
- The proposed approach is applied to the task of post summarization, by using the TL;DR dataset from Reddit
  - 3M messages + summaries from various subreddits
  - (TL;DR = Too Long; Didn't Read -- is a summary that is often added to long posts to provide a brief summary)
  - An LM can be fine-tuned to produce summaries of text

Collect human feedback

- Initial summaries are generated according to *some* model
  - E.g., models fine-tuned on different subsets of data
- A human annotator is presented with *pairs of summarises* for the same input
- The human chooses the *preferred summary*
  - **Note**: humans are better at picking the favorite between two items than they are at giving absolute grades
    - So, it's easier to say *Summary 1 is better than Summary 2*, then it is to say *Summary 1 has a score of 7.8*



Note
In Reinforcement Learning, a "policy" is a probability distribution across all possible actions, conditioned on the current state. For LMs, this corresponds to the output of a model, which is conditioned by the context. The policy, thus, is the probability distibution across all possible outputs

Humans as evaluation functions

- We can see the previous evaluation exercise as having a human that reads a post p, two summaries, s_1 and s_2 (in a text space C), and produces a verdict (0/1)

$$f(p, s_1, s_2) : C \times C \times C \rightarrow \{0, 1\}$$

- Or, we can explicitly assert that humans produce a "reward" for each summary via $r(\cdot)$, and produce a verdict 0/1 based on the largest reward

$$r(p, s): C \rightarrow \mathbb{R}$$

$$f(p, s_1, s_2) = \mathbf{1}(r(p, s_1) > r(p, s_2))$$

Cons of humans

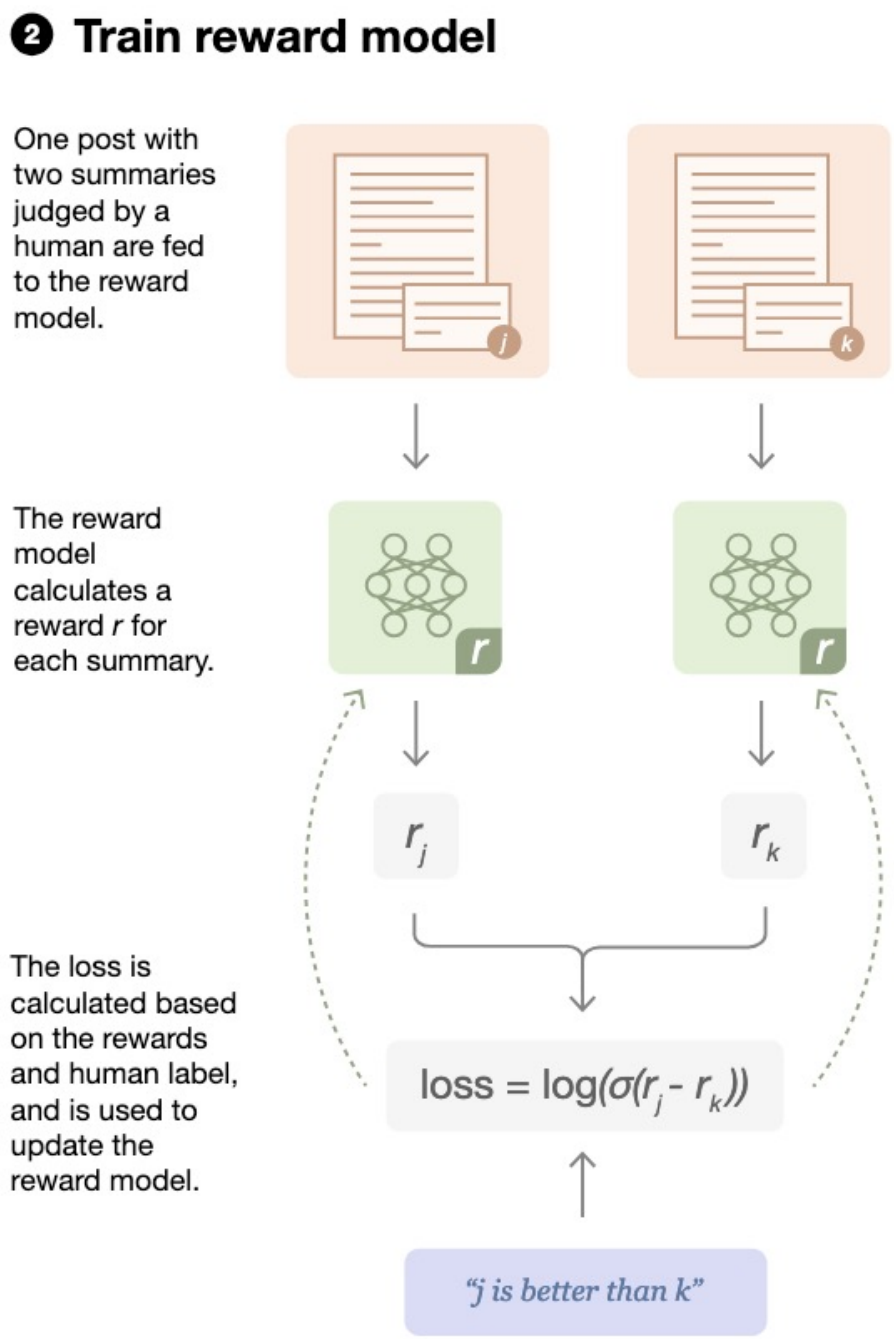
- *Cost & scalability*: collecting human feedback at the scale required for fine-tuning a model is impractical and expensive

- *Inconsistency*: different humans can be inconsistent. The same human may also be inconsistent across evaluations
  - (e.g., by saying that $r(a) > r(b)$ and $r(b) > r(c)$, but $r(c) > r(a)$)

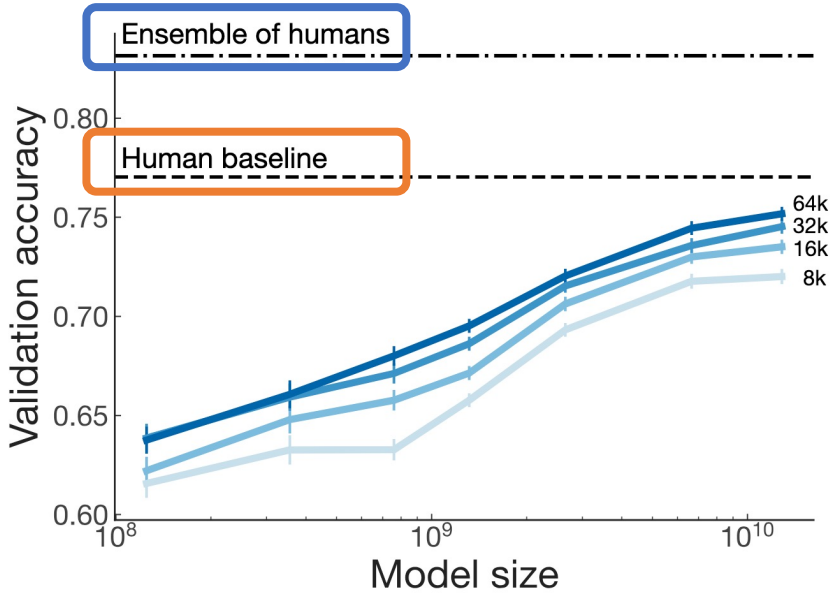
- *Simplicity of feedback*: specific annotations may consider only one or few aspects of interest. Producing a more sophisticated score that objectively accounts for various aspects may be difficult for humans to do

Train a Reward Model

- Instead of using human feedback directly, we train a *Reward Model* r_θ (another LM)
- The reward model predicts a *scalar value*, proportional to the quality of the result
- For two summaries s_j, s_k of a post p, we can produce the scores:
  - $r_j = r_\theta(p, s_j)$
  - $r_k = r_\theta(p, s_k)$
- We compute the loss of the reward model to *maximize the gap in rewards*:
  - For instance, $r_j - r_k$ should be large if s_j is the "better" result
  - $\log(\sigma(r_j - r_k))$ is used in the paper



How good is the Reward Model?



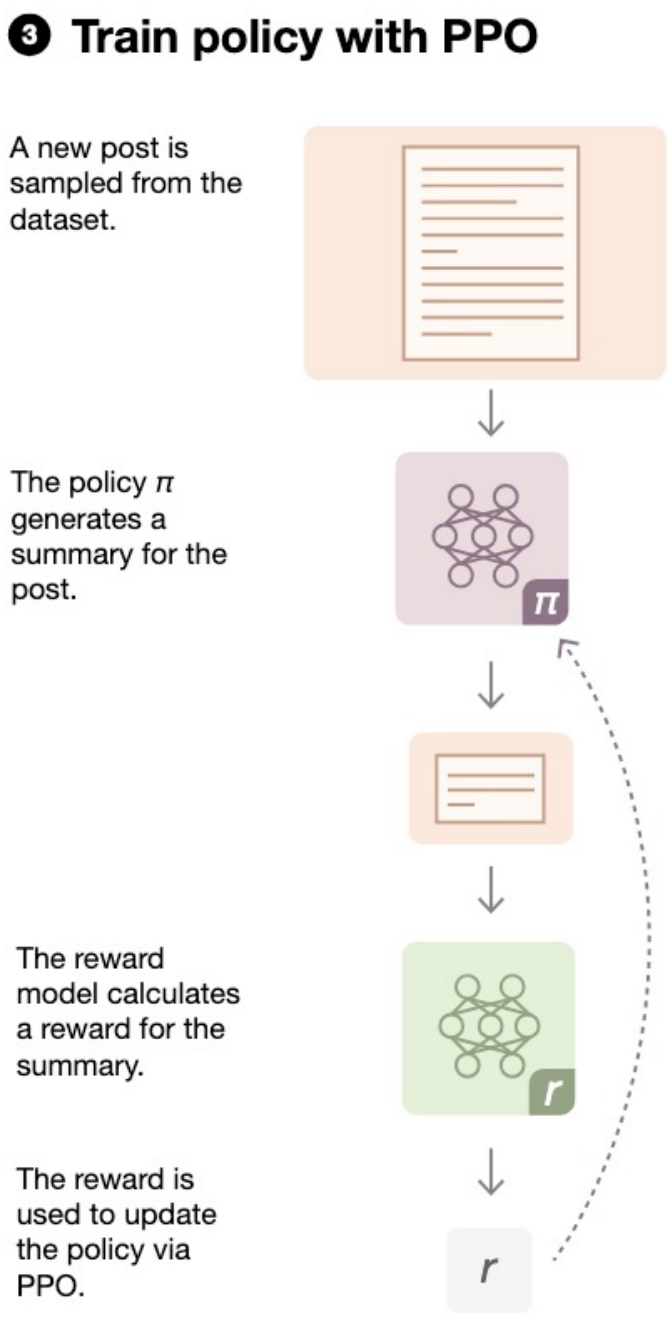
- One major concern of replacing humans with models is about *the quality of the result*
- *Can a model make the same predictions as humans?*
- For various Reward Model sizes (on the x axis) and various dataset sizes (colors), the authors measured if the RM can achieve human-level performance
- The results show:
  1. Larger models achieve better results (2x model → 1.8% increase)
  2. More annotated data improves results (2x data → ~1.1% increase)
  3. Results get close to the performance of a *single human*
  4. But, not as good as the *ensemble of humans* approach

Fine-tune model on feedback

- We make a *copy* of the *original* model (fine-tuned on the Reddit TL;DR dataset)
  - Let's refer to the *original policy* as π_{old}, and the one of the *copy* as π_{new}
  - We *fine-tune the copied model* that produces the copy policy

- We use the RM r_θ to produce the reward to give to the policy π_{new}
  - *"how much would a person like the answer?"*

Note
We call it a "reward" but has a similar role as a loss (in this case, we want to maximize it)



Fine-tuning on human feedback

- We generate a summary y based on the "copy" policy
  - In other words, y is sampled from π_{new}
- We can frame the reward for the model as:

$$R(x,y) = r_\theta(x,y) - \beta \log \left[\frac{\pi_{new}(y|x)}{\pi_{old}(y|x)} \right]$$

Note
This terms, referred to as a KL divergence (*), quantifies how distant two distributions are.

Intuitively, if π_{new} produces a sentence considered very unlikley by π_{old}, the ratio will be > 1, thus introducing a penalty.

In other words, it acts as a regularizer that prevents the model from producing outputs that are *too different* from the original model.

The β parameter controls the strength of the regularization.

Note
The main driver of the reward is the RM. Does π_{new} produce a good summary, according to the Reward Model? (*will a human like it?*)

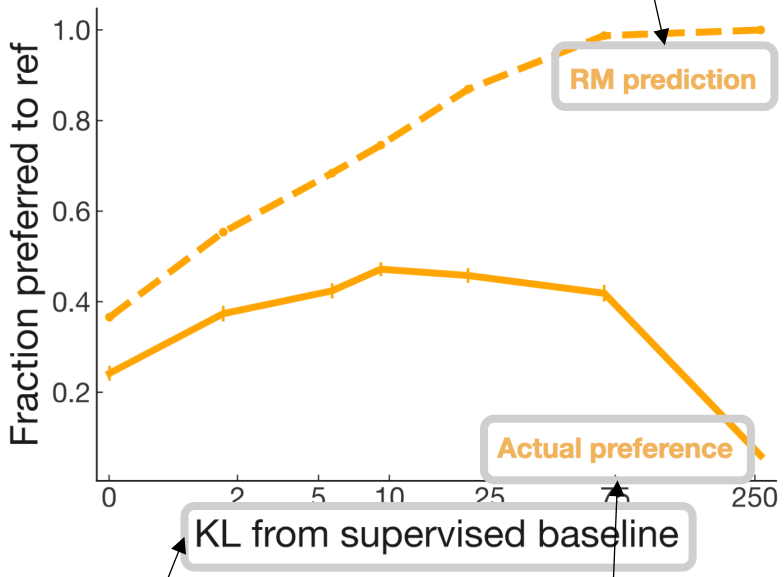
(*) The KL (Kullback-Leibler) divergence is actually defined in a slightly different way. However, it still measures how dissimilar two probabilities distributions are. The log-ratio is one part of the KL divergence.

We also use a clipping on the probability ratio. This prevents, at any step, updating the model "too much". This clipped log probability ratio is part of the contributions of PPO (Proximity Policy Optimization), a Reinforcement Learning technique that allows achieving better training stability and sample efficiency

Why use a "KL divergence"?

- It is tempting to maximize the reward $r(\cdot)$, regardless of how distant from the original model we go
- However, $r(\cdot)$ is a *proxy for human preference*, not the actual human preference
- Maximizing $r(\cdot)$ ends up producing a very different from the original one
  - The new model improves the score from the reward model, but it no longer provides useful summaries

What the *reward model* thinks should happen (the summaries should be great, if we get away from the original model)



How distant the model is from the *original (fine-tuned) model*

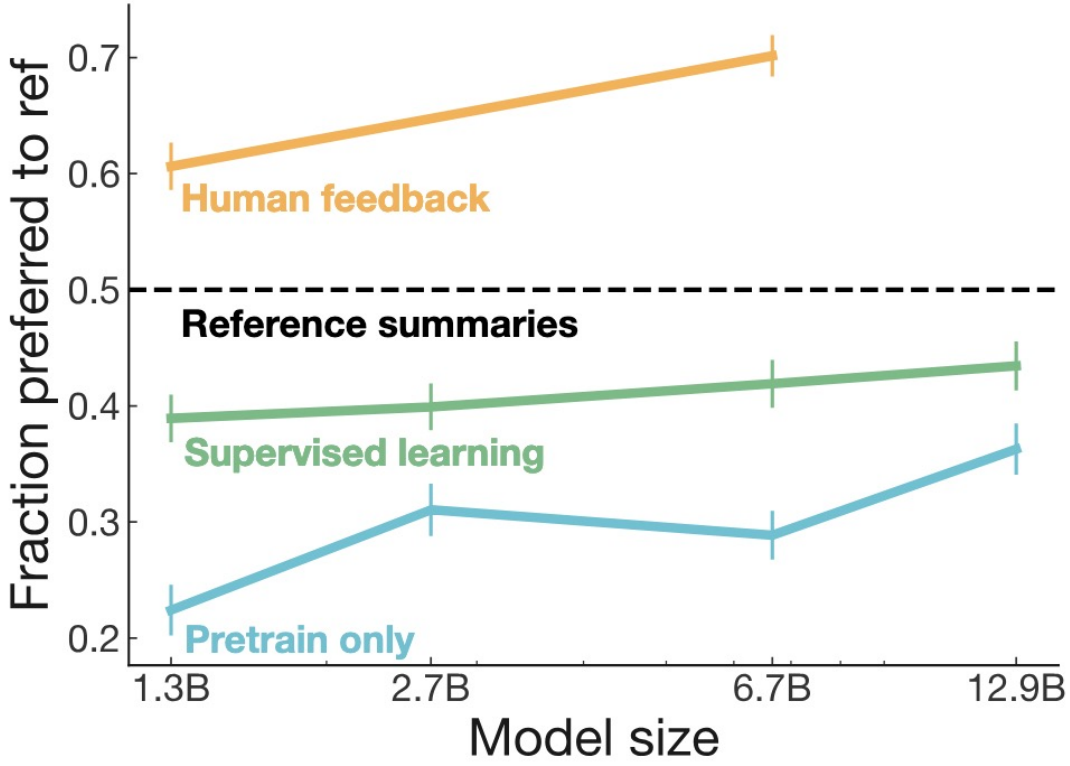
However, humans end up not liking those highly divergent summaries (the *fine-tuned model* was producing "acceptable" versions after all)

Preference results

- Annotators are asked to choose the *preferred version* between the human-written and the model-generated summary

- Different models are compared:
  - *Pretrain only*: the base LM, without fine-tuning
  - *Supervised learning*: the LM, fine-tuned on the Reddit TL;DR dataset
  - *Human feedback*: the *supervised learning* LM, additionally fine-tuned to improve based on the human feedback reward

- Results show a *clear human preference for the HF model* w.r.t. all others
  - even better than the human-generated!



Aligning instruction-tuned models

- The previous work was focused on a *single task* (summarization)
  - Other works explored alignment for other, single tasks
- Ouyang et al., 2022 (OpenAI) introduces *InstructGPT*
  - Extending the model to address various tasks
  - Using *instruction tuning*, using human-written answers (not with templating!)
- Main takeaways:
  - Annotators prefer InstructGPT (1.3B) outputs over GPT-3 (175B)
  - InstructGPT is *more truthful*, slightly *less toxic* than GPT-3
  - InstructGPT is aligned to annotators it never learned from
  - InstructGPT *generalizes to new tasks* not in the fine-tuning datasets

Ouyang, Long, et al. "Training language models to follow instructions with human feedback." *Advances in neural information processing systems* 35 (2022): 27730-27744. https://arxiv.org/pdf/2203.02155

InstructGPT steps

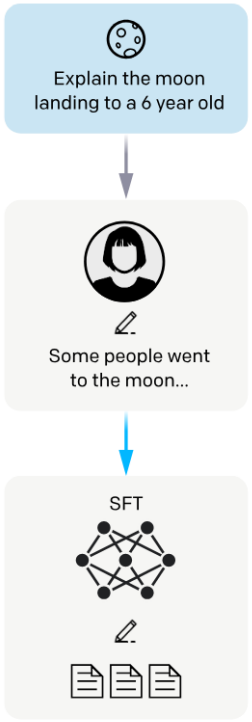
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



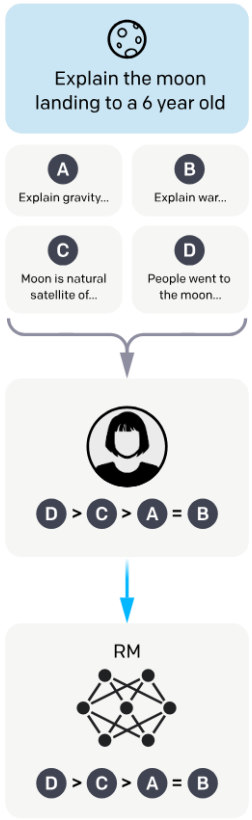
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

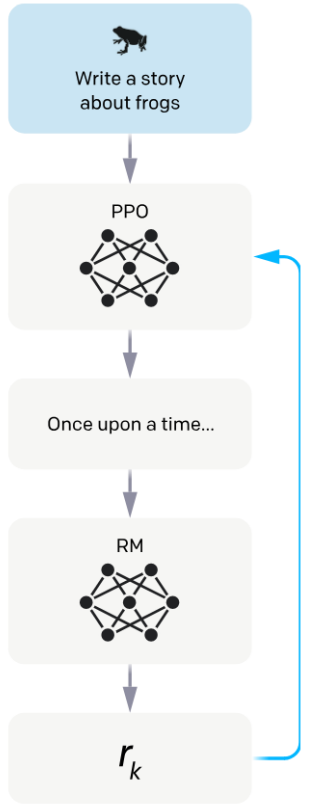
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



A new trend in town

- InstructGPT shows that:
  - Pretraining alone is not enough for user-aligned behavior.
  - Fine-tuning using *high-quality, instruction-following* data became essential.
- New Approach:

  1. *Pretrain* a model on *large quantities* of (dirty) data
     - Scraped from various, potentially unreliable, sources
     - This stage focuses on learning general language patterns and knowledge
     - These models exhibit alignment issues with user instructions or generating high-quality, useful outputs
  2. *Collect* smaller, *higher-quality datasets*
     - Gather instruction-based datasets where human feedback refines the model's responses.
  3. Use RLHF to *align models* to user preferences
- The original ChatGPT is based on InstructGPT (w/GPT-3 175B)