CSCI 5541: Natural Language Processing

Lecture 14: Instructing and augmenting LLMs

Zae Myung Kim
https://zaemyung.github.io/

computer science
& engineering

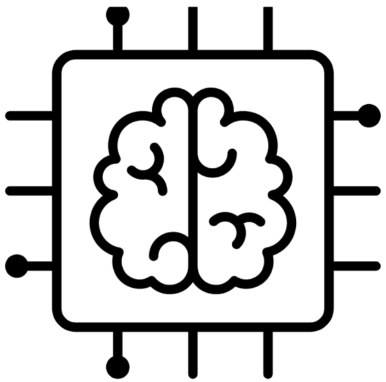
MINNESOTA NLP EST. 2021

University of Minnesota
Driven to Discover®

Era of Large Language Models (LLMs)

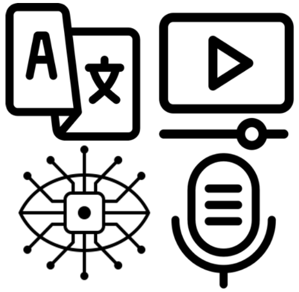
Multi-task generalization

Generative Search



LLM

Human preferable responses



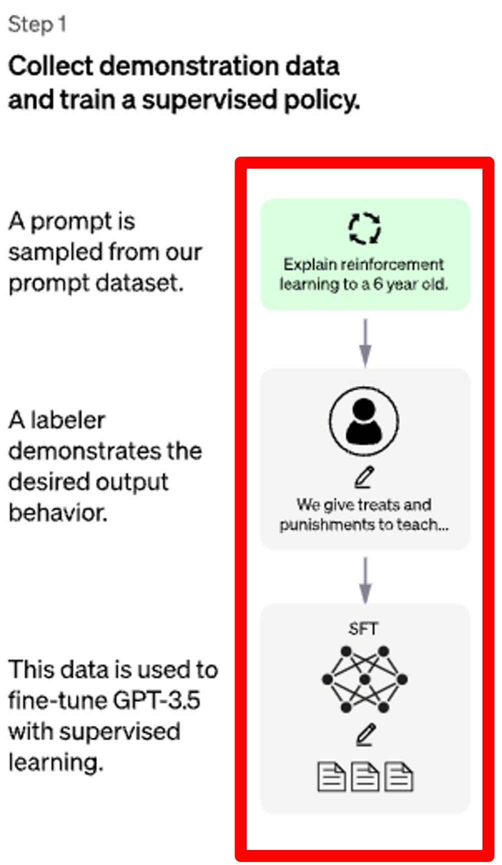
Multi-lingual, Multi-modal

Generative Agents

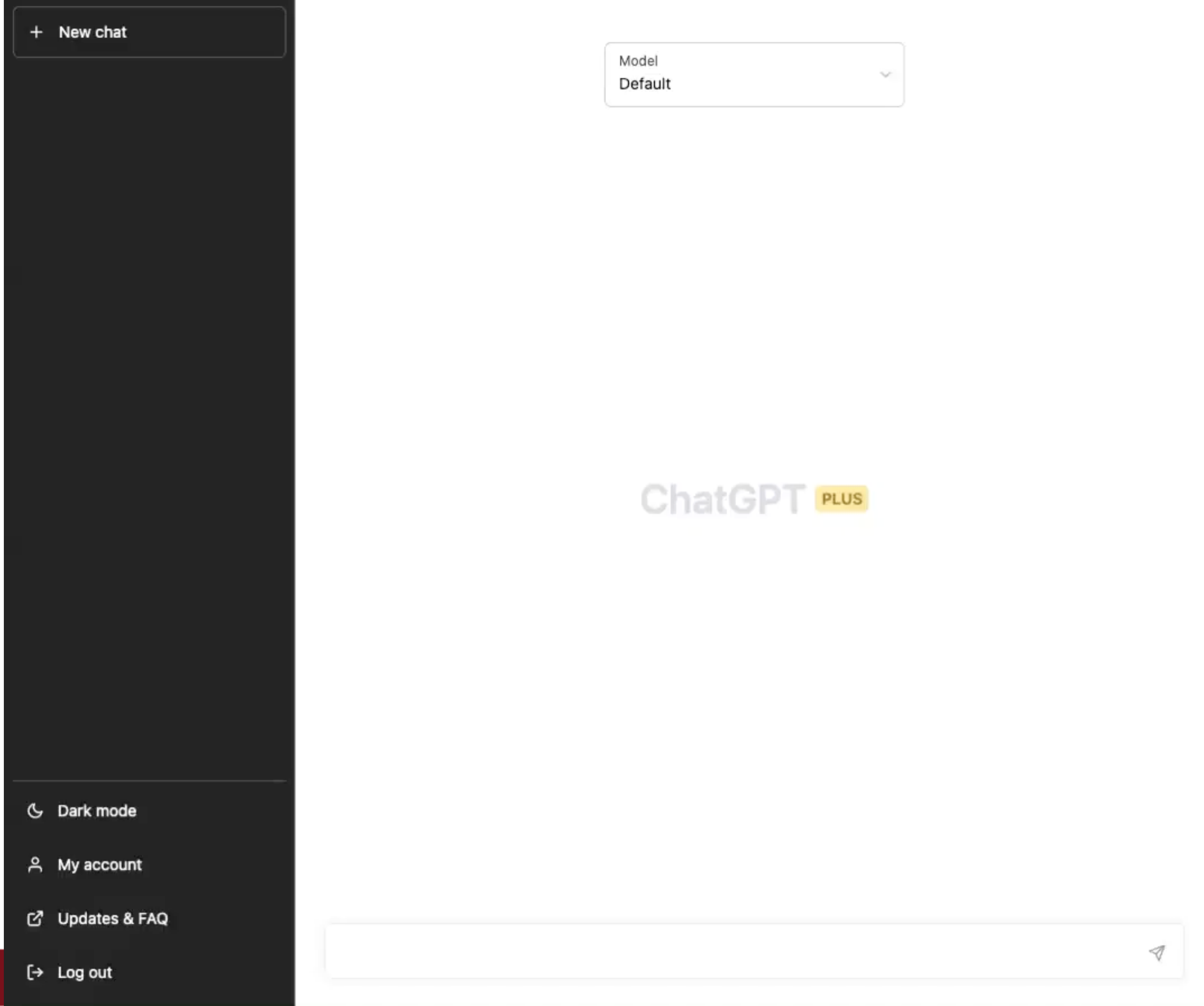
Tool Integration

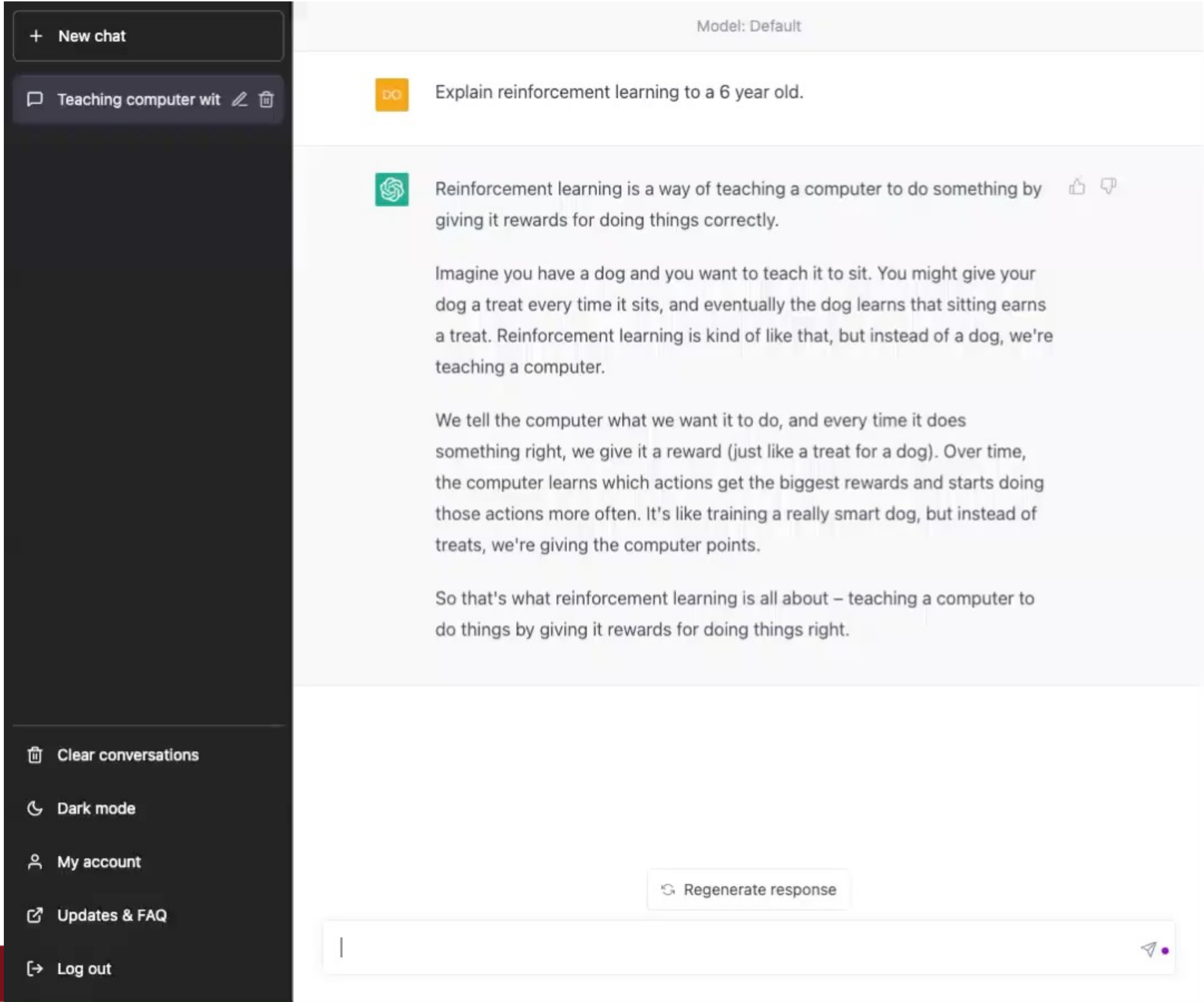
Brief Overview of ChatGPT Training

Training details in ChatGPT

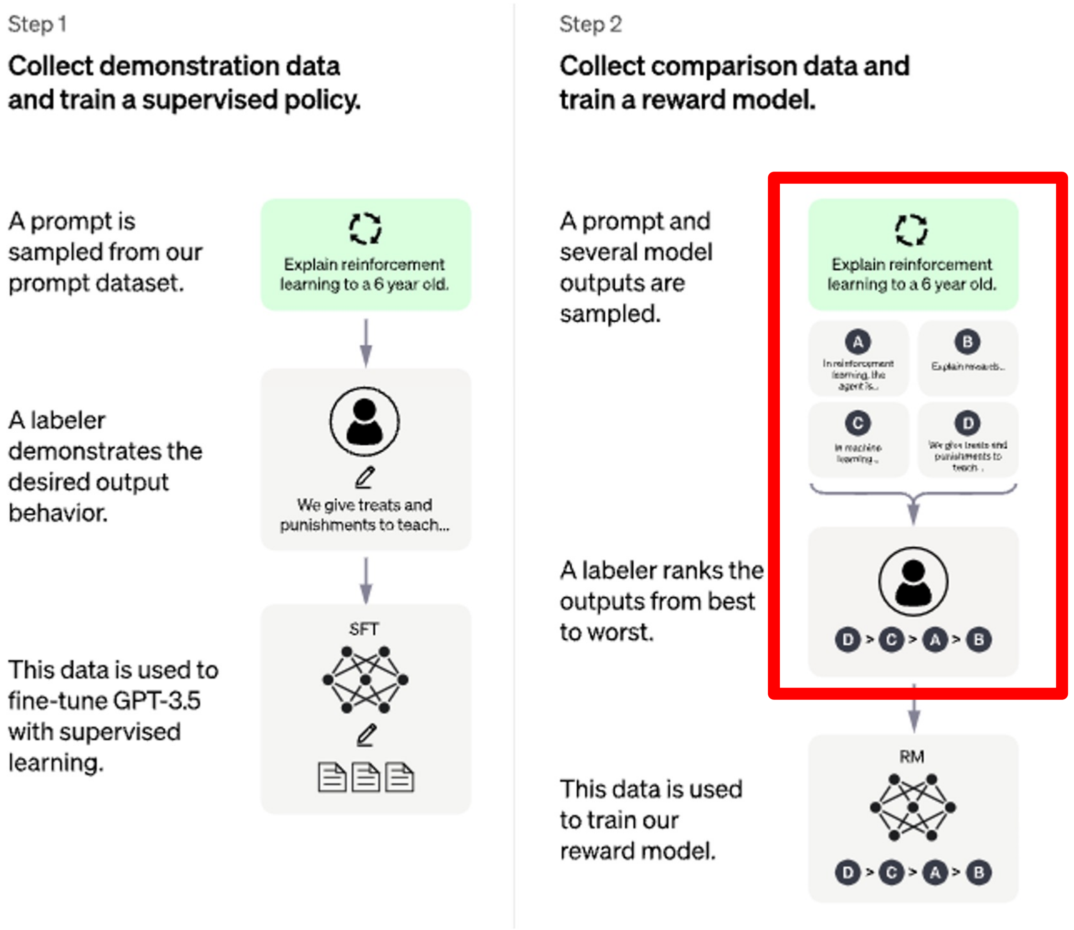


Instruction Tuning
(Supervised Finetuning)


New chat
Model
Default
ChatGPT PLUS
Dark mode
My account
Updates & FAQ
Log out


New chat
Teaching computer wit
Clear conversations
Dark mode
My account
Updates & FAQ
Log out
Model: Default
DO
Explain reinforcement learning to a 6 year old.
Reinforcement learning is a way of teaching a computer to do something by giving it rewards for doing things correctly.
Imagine you have a dog and you want to teach it to sit. You might give your dog a treat every time it sits, and eventually the dog learns that sitting earns a treat. Reinforcement learning is kind of like that, but instead of a dog, we're teaching a computer.
We tell the computer what we want it to do, and every time it does something right, we give it a reward (just like a treat for a dog). Over time, the computer learns which actions get the biggest rewards and starts doing those actions more often. It's like training a really smart dog, but instead of treats, we're giving the computer points.
So that's what reinforcement learning is all about – teaching a computer to do things by giving it rewards for doing things right.
Regenerate response

Training details in ChatGPT



Reward Models (Preference Modeling)

Example human reward or preference

Moving Table to Left and Adding Space to Columns

Ask Question

Asked 11 years ago Modified 5 years, 7 months ago Viewed 8k times

3

In the following table, the numbers take up too much room. I do not want to landscape the table because there are too many rows.

So, I'd like to do both of the following:

1. Create more space between some of the columns. Especially if some columns don't require as much width, I'd like to know whether it is better that I manually set this space in order to optimize.
2. Shift the table to both the left and right in the process. When I adjust the `\textwidth`, that just moves it to the right. I'd like the table to still remain centered within the page.

I'd like for the contents to be centered within each column as they are with the current code.

I see a similar posting, but I don't know how to incorporate the results and if that is exactly what I'm trying to do: Adding space between columns in a table

```
\documentclass[12pt,english]{article}
\usepackage{longtable}
\usepackage{fullpage}
\usepackage{times}
\usepackage[flushleft]{threeparttable}
\usepackage[font=large,labelfont=bf,tableposition=top,text
\usepackage{tabularx}
\usepackage{booktabs}
\newcolumntype{C}{>{\centering\arraybackslash}X}

\begin{document}
```

The Overflow Blog

- Why governments need open source more than ever
- Stop saying "technical debt"

Featured on Meta

- Ticket smash for [status-review] tag: Part Deux
- We've added a "Necessary cookies only" option to the cookie consent popup

Linked

- 217 Adding space between columns in a table
- 41 Change \textwidth and \textheight in mid-document
- 7 Trying to replicate a table from academic paper

Related

- 17 How to format table with long column head entries?
- 217 Adding space between columns in a table
- 4 How to center numbers in a table column with the siunitx package?

Answer A is "better" than Answer B

6

Here are my suggestions:

- Insert the command `\setlength\tabcolsep{3pt}` to cut the amount of intercolumn whitespace in half. (The default value of this parameter is `6pt`.)
- Replace the instruction `\small` (immediately after `\begin{threeparttable}`) with `\footnotesize`. (This also lets you get rid of the subsequent `\footnotesize` command.)
- Eliminate the vertical whitespace before the first column and after the final column by changing the `tabularx` setup as follows:

```
\begin{tabularx}{\textwidth}{@{}l*{10}{C}@{}}
```

(note the two new `@{}` elements).

With these changes, I manage to get the table to fit into the allocated textblock width. My papersize is `US Letter`; if yours is `A4`, you'll probably need to reduce the `tabcolsep` macro's value further, to `2pt`.

2

In addition to what @mico said I don't see why you want to use tabularx here. Your example fitted within the measure if i changed it to

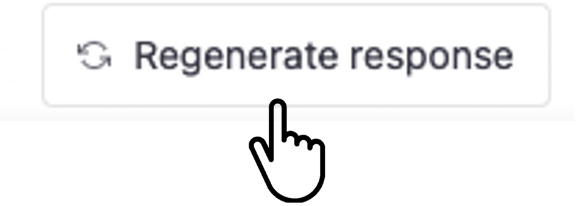
```
\setlength\tabcolsep{1pt}
\begin{tabular}{l*{10}{l}}
```

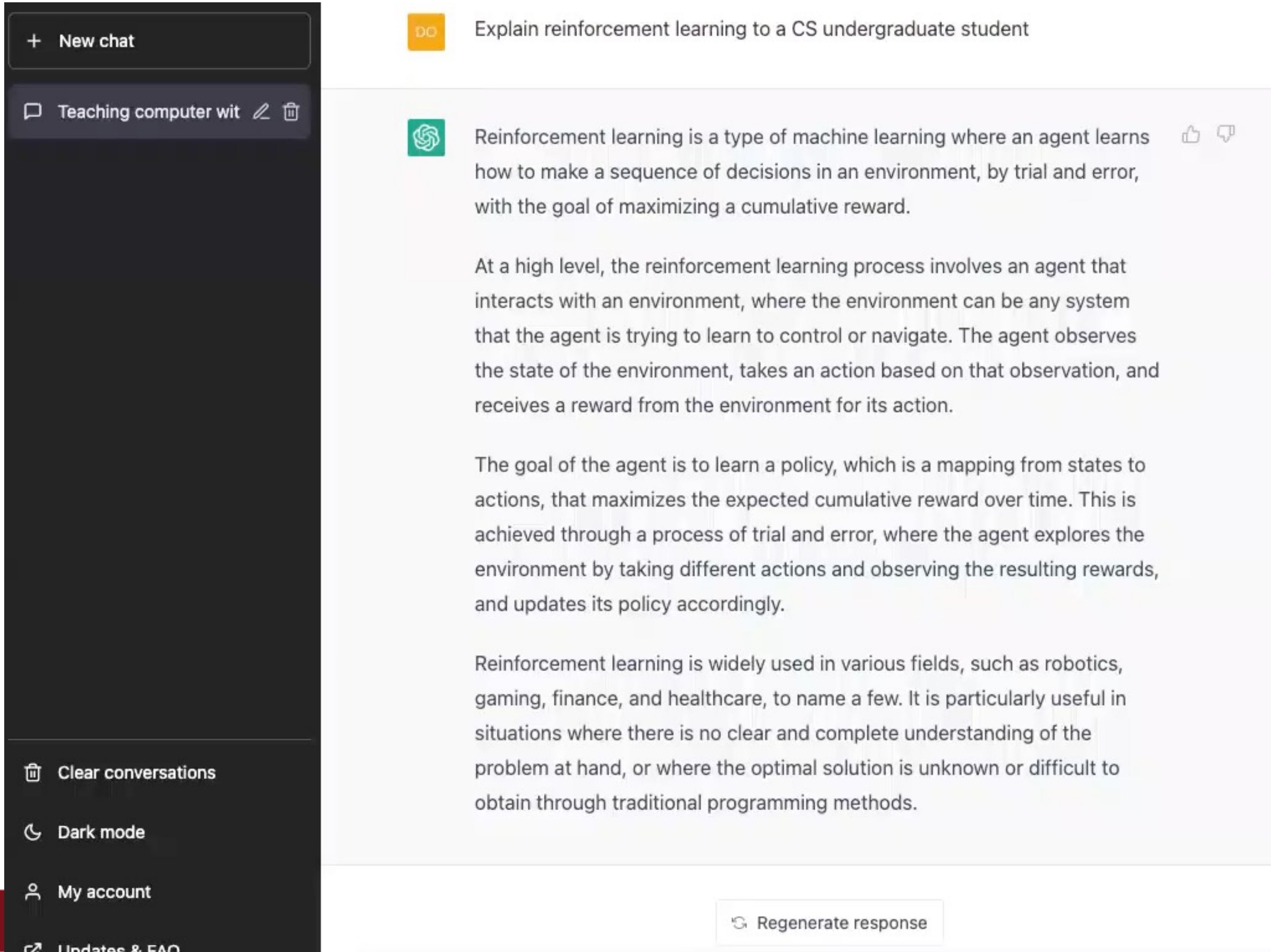
If you really need to extend into the margins just for one table then the plain tex `\centerline` is a quick and easy way of doing so.

```
\centerline{\begin{tabular}{l*{12}{l}}
\hline \hline \addlinespace
 & (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (1
Variable Name & 1234566 & 6543216 & 2233456 & 6655432 & 183
\hline \hline \addlinespace
 \end{tabular}}
```

Share Improve this answer Follow edited Jul 18, 2017 at 8:48

Human preference from demonstration





Human preference from demonstration

"Explain reinforcement learning to a CS undergraduate student"

Reinforcement learning is a type of machine learning where an agent learns how to make a sequence of decisions in an environment, by trial and error, with the goal of maximizing a cumulative reward.
At a high level, the reinforcement learning process involves an agent that interacts with an environment, where the environment can be any system that the agent is trying to learn to control or navigate. The agent observes the state of the environment, takes an action based on that observation, and receives a reward from the environment for its action.
The goal of the agent is to learn a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward over time. This is achieved through a process of trial and error, where the agent explores the environment by taking different actions and observing the resulting rewards, and updates its policy accordingly.
Reinforcement learning is widely used in various fields, such as robotics, gaming, finance, and healthcare, to name a few. It is particularly useful in situations where there is no clear and complete understanding of the problem at hand, or where the optimal solution is unknown or difficult to obtain through traditional programming methods.

R1

Reinforcement learning is a type of machine learning that involves training an agent to make decisions based on feedback received from its environment. The goal is to have the agent learn to take actions that will maximize some long-term reward.
The agent interacts with its environment by taking actions and receiving feedback in the form of a reward signal. The reward signal is used to evaluate the goodness of the agent's actions, with the goal of increasing the likelihood of receiving high rewards in the future.
The agent learns through trial and error, using past experiences to inform future decision making. By considering the rewards associated with different actions, the agent can develop a policy that will lead to the best possible outcome over time.
One of the key challenges in reinforcement learning is striking a balance between exploration and exploitation. The agent needs to explore new actions to discover potentially high-reward actions, but also needs to exploit actions that are already known to be effective.
Reinforcement learning has been successfully applied in a wide range of areas, including robotics, gaming, and finance. It has also been used to develop advanced decision-making systems, such as self-driving cars and intelligent virtual assistants.

R2

Reinforcement learning is a subfield of artificial intelligence that involves training an agent (e.g., a software agent, a robot, or a game character) to make decisions in a complex, dynamic environment. In reinforcement learning, the agent learns by receiving feedback in the form of rewards or punishments for its actions, without being explicitly told which actions to take.
The agent interacts with the environment, taking actions based on its current state, and the environment responds by providing feedback in the form of a numerical reward signal that reflects how well the agent is doing. The agent's objective is to learn a policy, which is a mapping from states to actions, that maximizes its cumulative reward over time.
Reinforcement learning algorithms use various techniques, such as value functions, policy gradients, and Q-learning, to optimize the agent's policy. These algorithms typically involve a trade-off between exploration (trying out new actions to learn more about the environment) and exploitation (taking actions that the agent already knows to be good).
Reinforcement learning has applications in a wide range of domains, including robotics, gaming, recommendation systems, and self-driving cars.

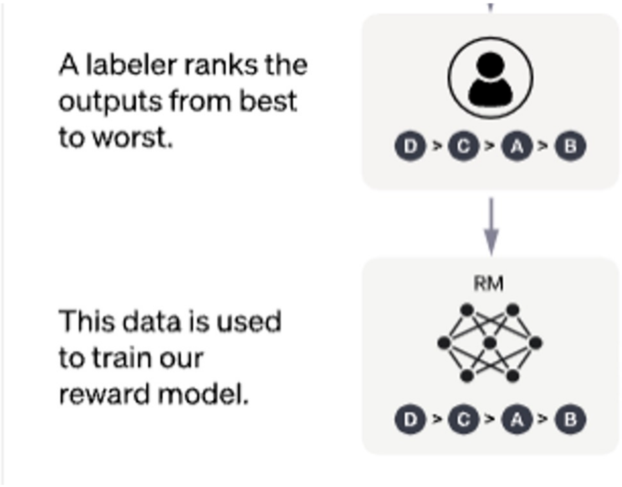
R3

Reinforcement learning is a type of machine learning that enables a computer to learn how to make decisions through trial and error.
In reinforcement learning, an agent interacts with an environment by taking actions and observing the resulting state and reward. The agent's goal is to learn a policy, or a mapping from states to actions, that maximizes the expected cumulative reward over time.
The agent starts off knowing nothing about the environment or the best actions to take, but it learns by taking actions and observing the resulting rewards. By repeatedly exploring and exploiting the environment, the agent gradually learns a good policy that enables it to maximize its cumulative reward.
Reinforcement learning has been successfully applied in a variety of domains, such as game playing, robotics, and recommendation systems. It is a powerful approach for learning how to make decisions in complex and uncertain environments.

R4

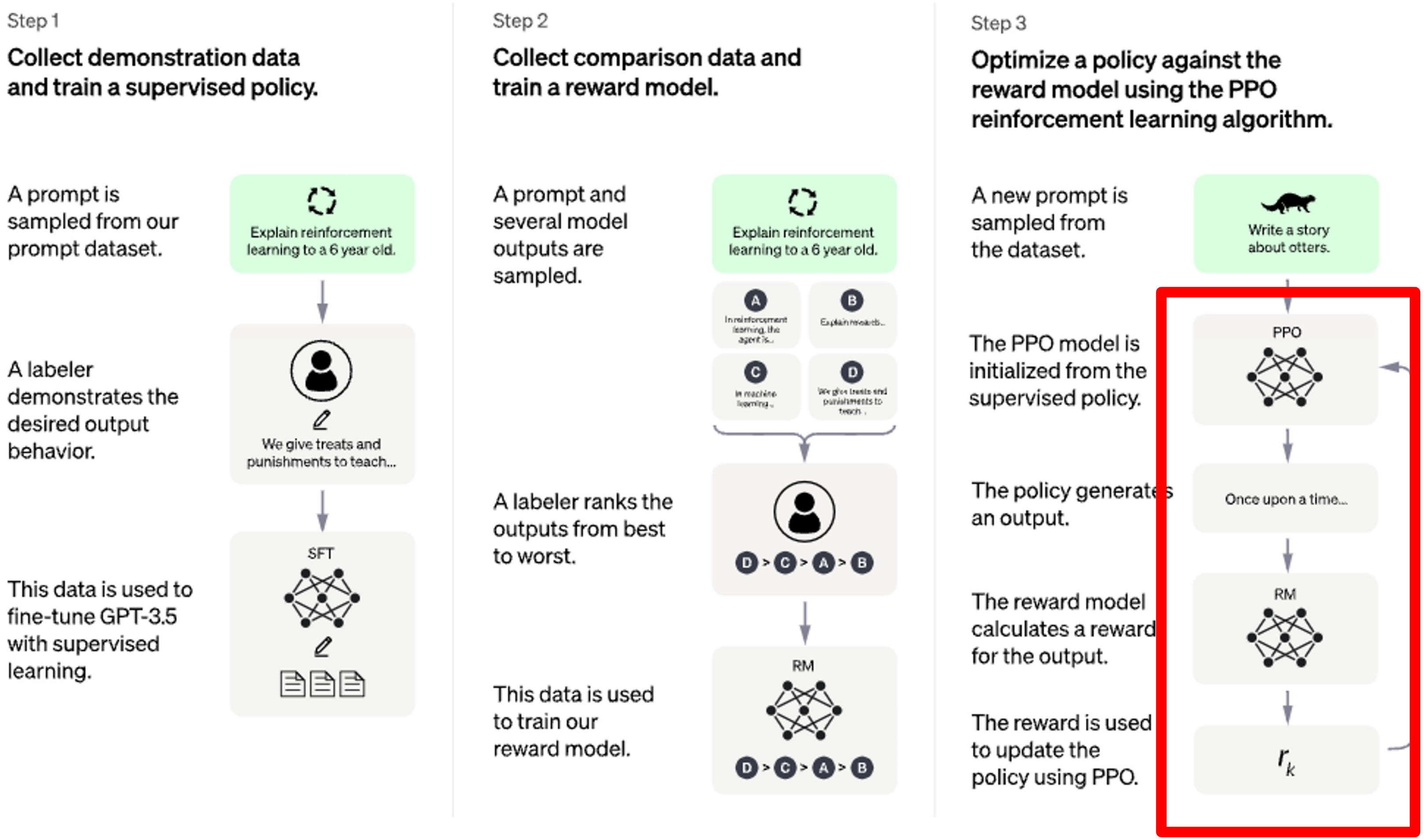
R3 > R2 > R4 > R1

Preference/Ranking as a good answer



Training details in ChatGPT

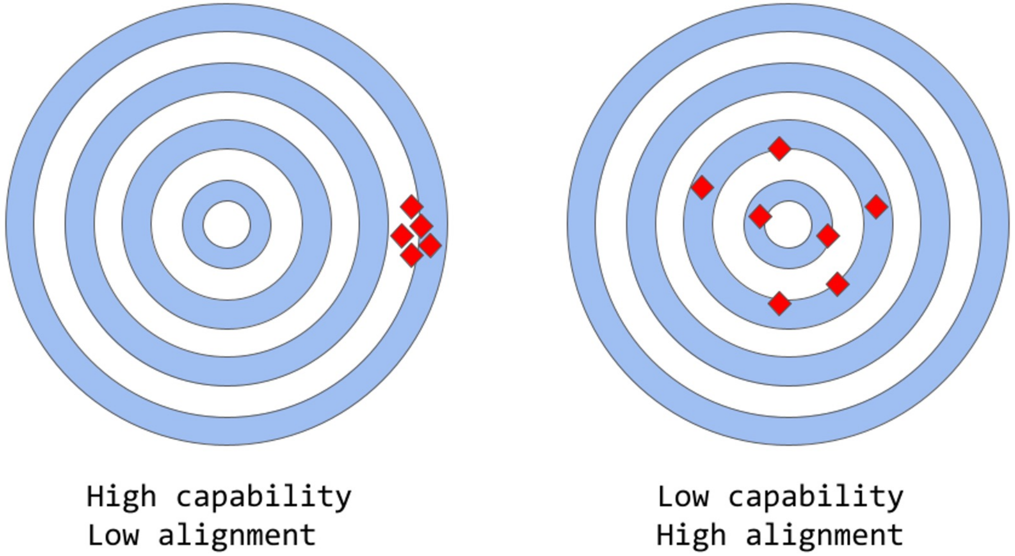
Reinforcement Learning with Human Feedback (RLHF)



Human Alignment

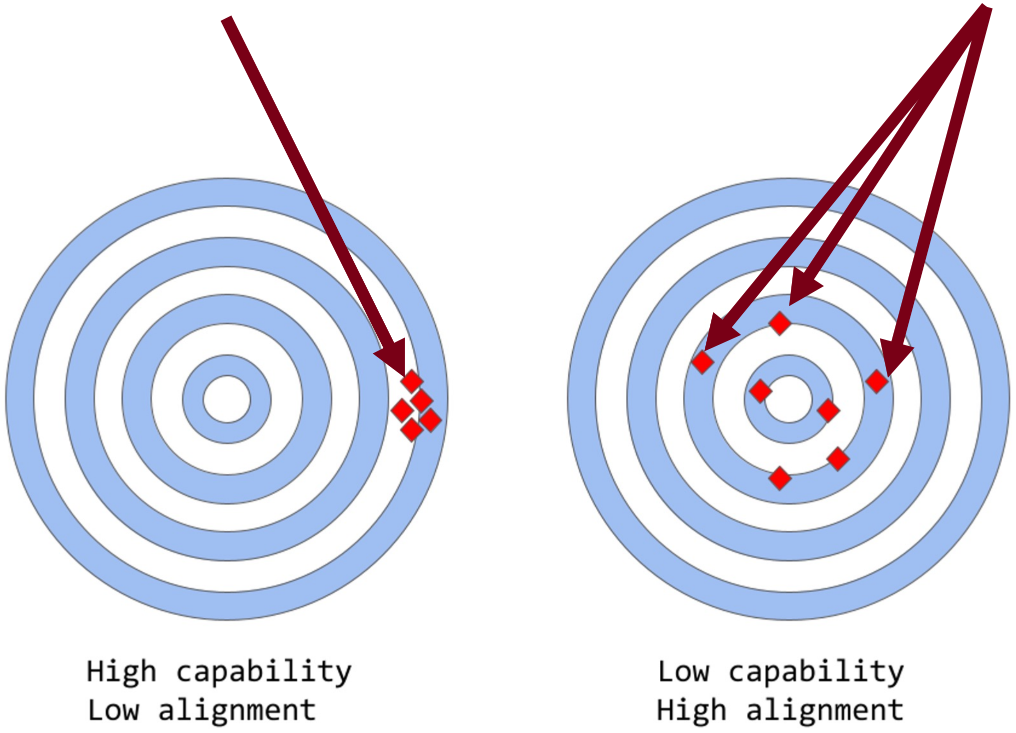
Human Alignment

- A model's **capability** is typically evaluated by how well it is able to optimize its objective function
- **Alignment** is concerned with what we actually want the model to do versus what it is being trained to do.



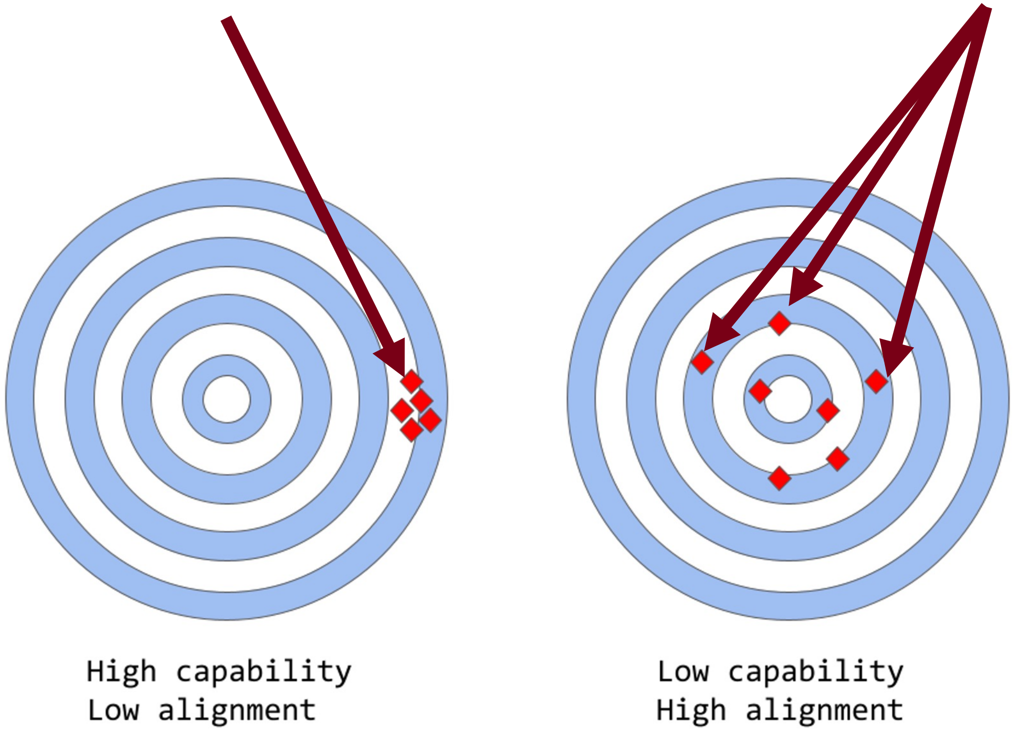
What we don't expect from LLMs:
- ❑ Lack of helpfulness
- ❑ Hallucinations
- ❑ Lack of interpretability
- ❑ Generating biased or toxic output

Objective: next or masked token prediction



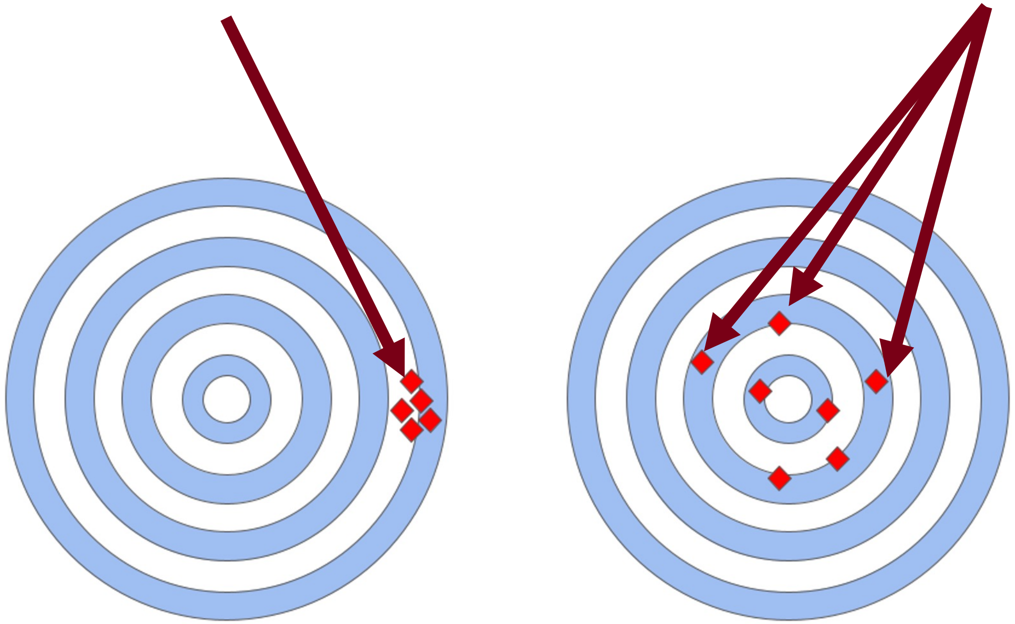
Objective: next or masked token prediction

What we don't expect from LLMs:



Objective: next or masked token prediction

What we don’t expect from LLMs:



High capability
Low alignment

Low capability
High alignment

Instruction Tuning

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

Language models are not aligned with user intent [Ouyang et al., 2022].

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION **Human**

A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.

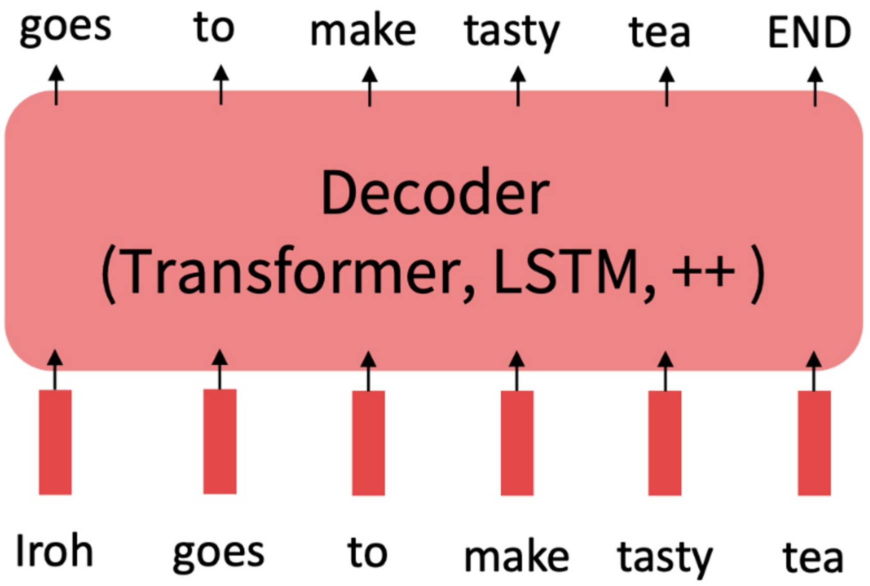
Language models are not aligned with user intent [Ouyang et al., 2022].

→ We can *finetune* it with responses we want!

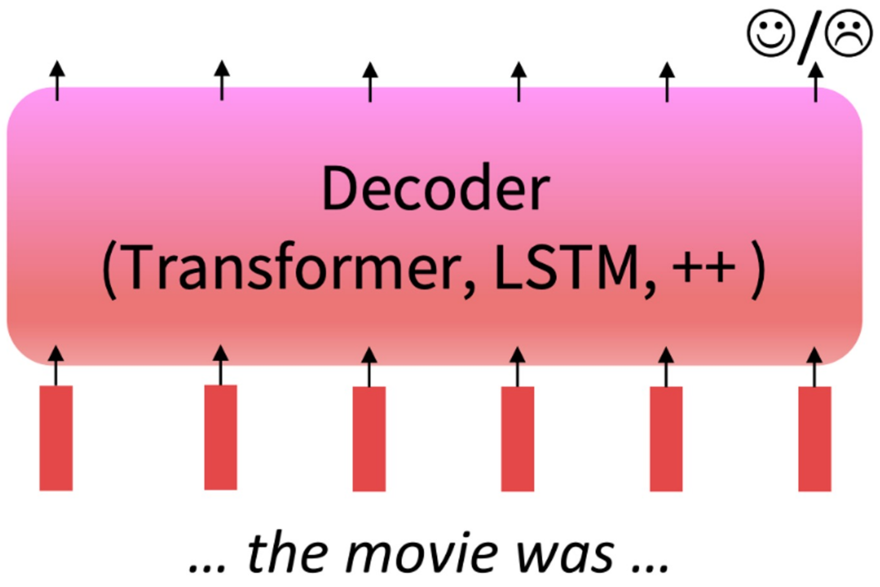
Recap on pretrain-finetune paradigm

- Pretraining can greatly improve performances on downstream NLP tasks by serving as parameter initialization.

Step 1: Pretrain (via language modeling)
Lots of text; learn general things!



Step 2: Finetune (on your task)
Not many labels; adapt to the task!

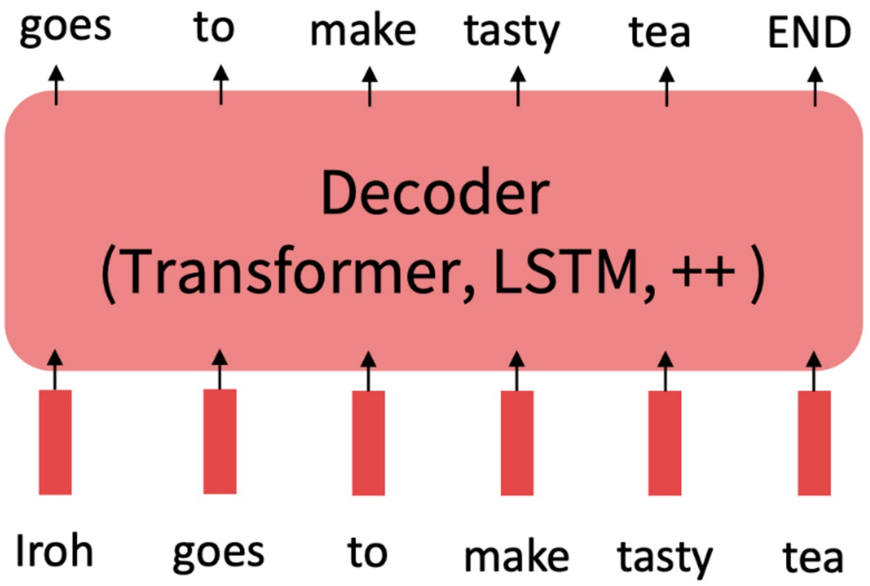


Source: cs224n, Stanford

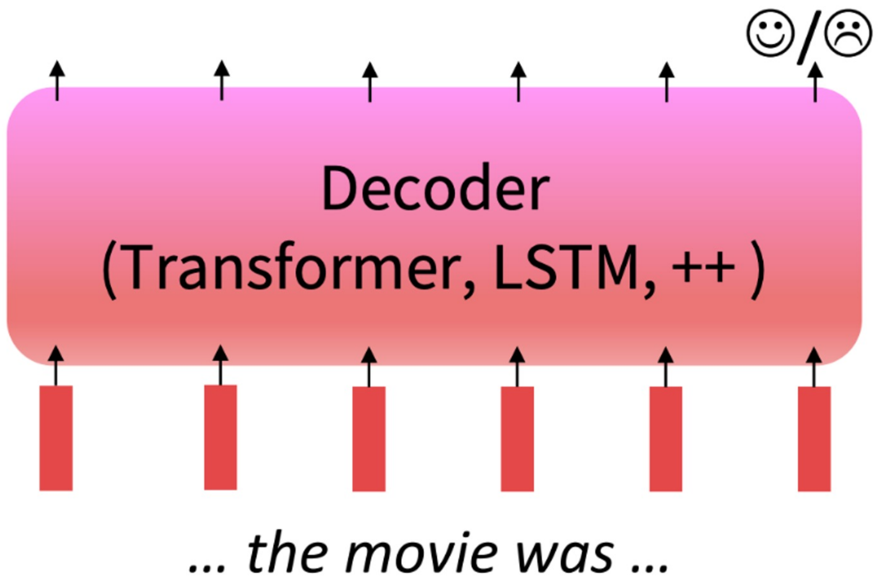
Scaling up finetuning

- Pretraining can greatly improve performances on downstream NLP tasks by serving as parameter initialization.

Step 1: Pretrain (via language modeling)
Lots of text; learn general things!



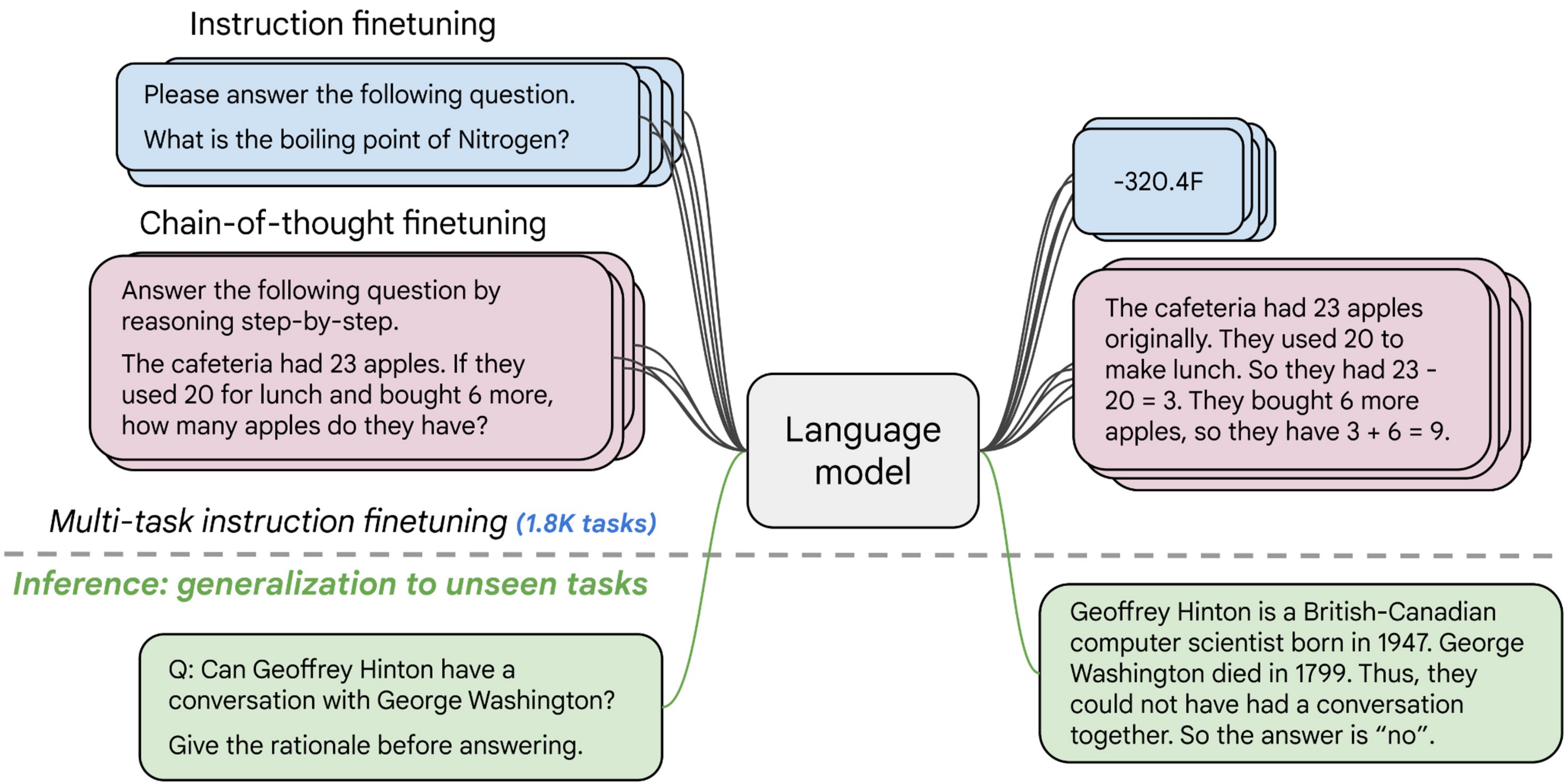
Step 2: Finetune (on many tasks)
Many labels; adapt to many tasks!



Source: cs224n, Stanford

Instruction finetuning

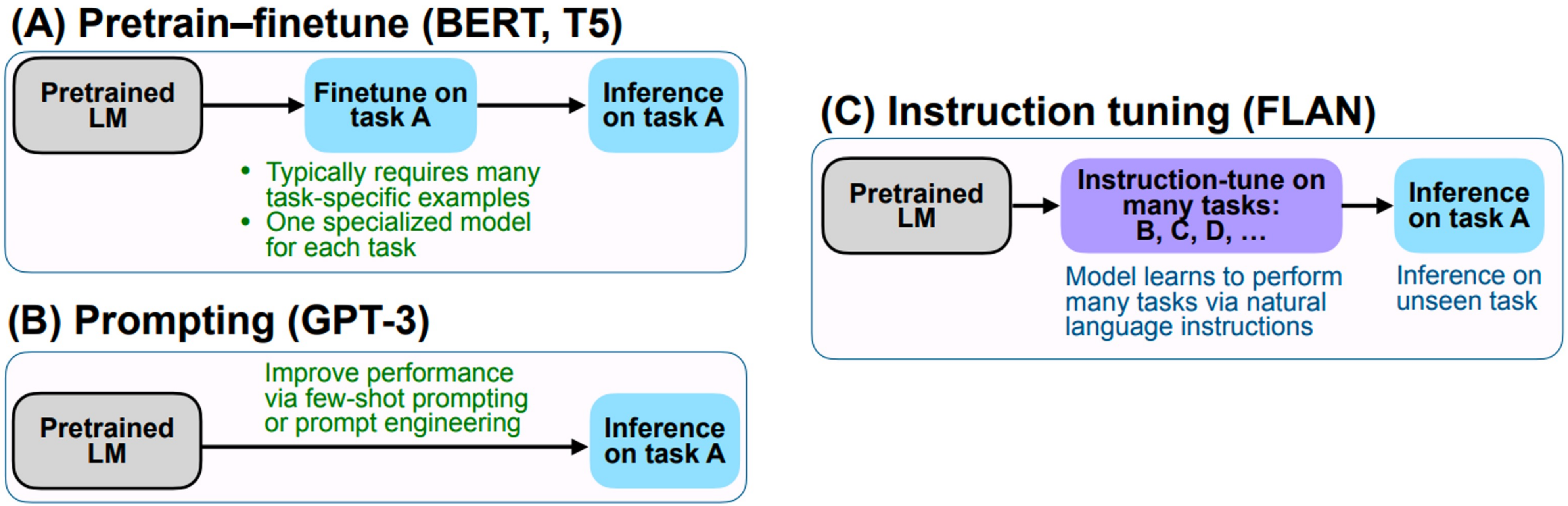
- Collect examples of ((instruction, input), output) pairs across many tasks and finetune an LM and evaluate on unseen tasks



[FLAN-T5; Chung et al., 2022]

Instruction finetuning vs. standard finetuning

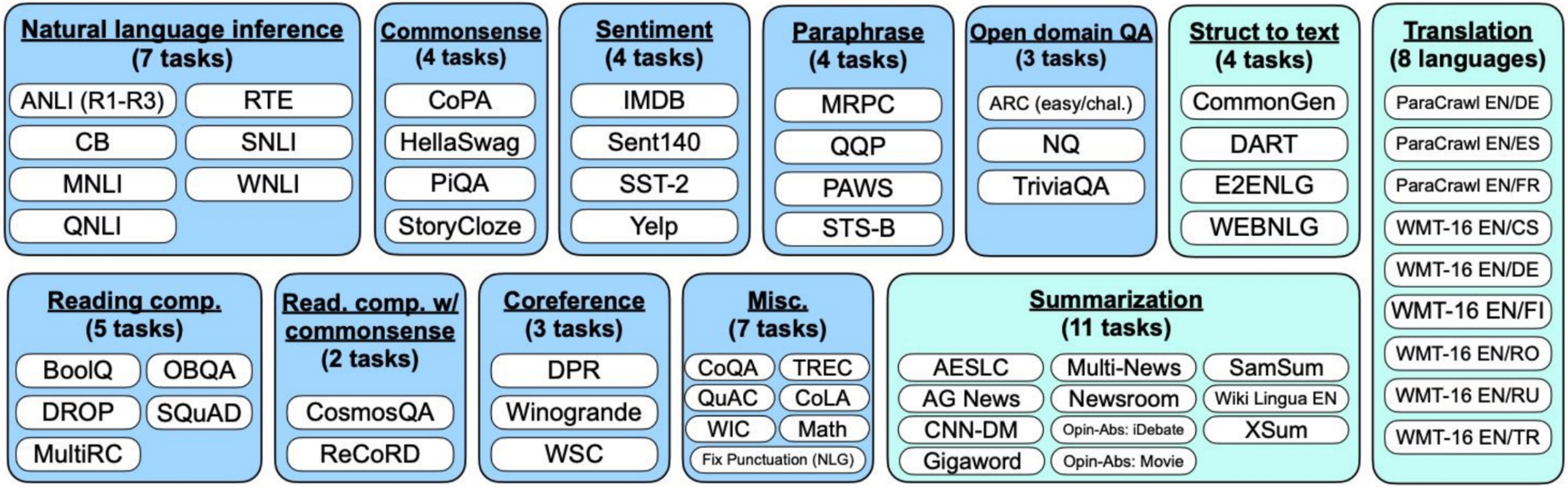
- The main difference lies in the data that the model is trained on
  - Standard supervised finetuning trains models on ***input examples*** and their ***corresponding outputs***.
  - Instruction finetuning augments ***input-output examples*** with ***instructions***, which enables instruction-tuned models to generalize more easily to new tasks.



[Wei et al., 2022]

Instruction finetuning

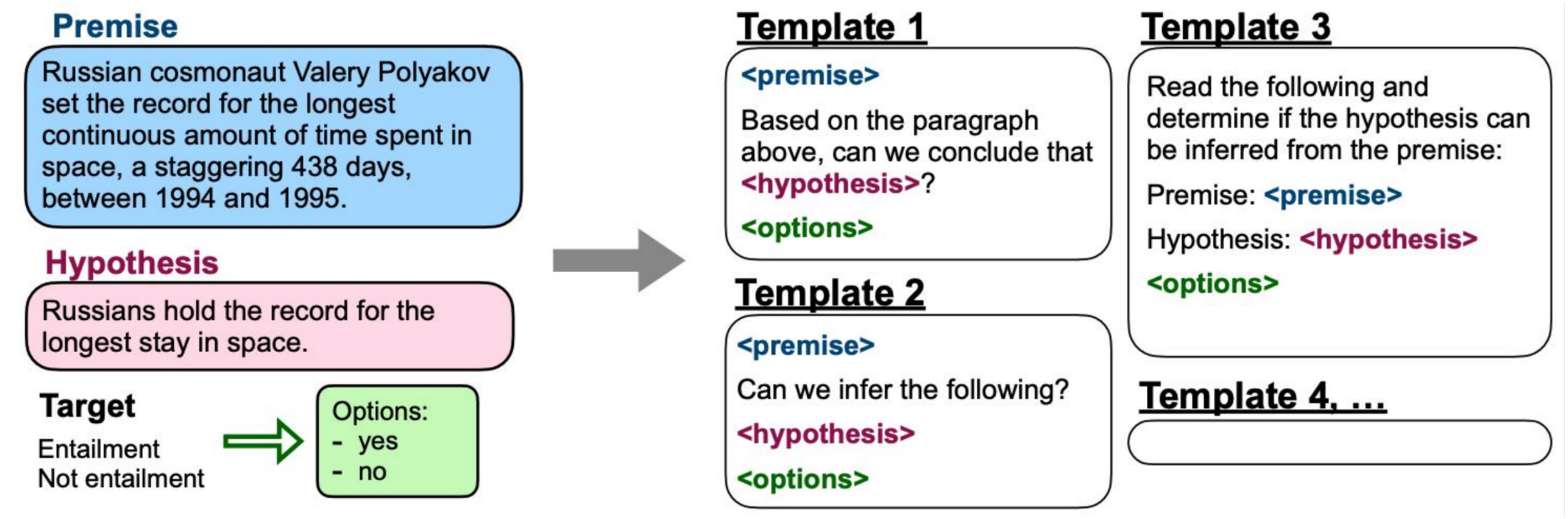
- 62 NLP datasets
- 12 “task clusters”



[FLAN-T5; Chung et al., 2022]

Instruction finetuning templates

- Natural instruction templates for each task



[FLAN-T5; Chung et al., 2022]

Instruction pretraining?

- Scaling up data and model improves performance.
- **SuperNaturalInstructions** dataset contains **over 1.6K tasks, 3M+ examples**

[Wang et al., 2022]

Or less is more?

- *"LIMA – Less Is More for Alignment"* (Zhou et al. 2023)
- Authors report that LLaMa 65B model finetuned on a collection of high quality and diverse 1,000 samples are enough to beat models trained on much larger instruction datasets.

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334
Dev			
Paper Authors (Group A)	50	36	N/A
Test			
Pushshift r/AskReddit	70	30	N/A
Paper Authors (Group B)	230	31	N/A

Table 1: Sources of training prompts (inputs) and responses (outputs), and test prompts. The total amount of training data is roughly 750,000 tokens, split over exactly 1,000 sequences.

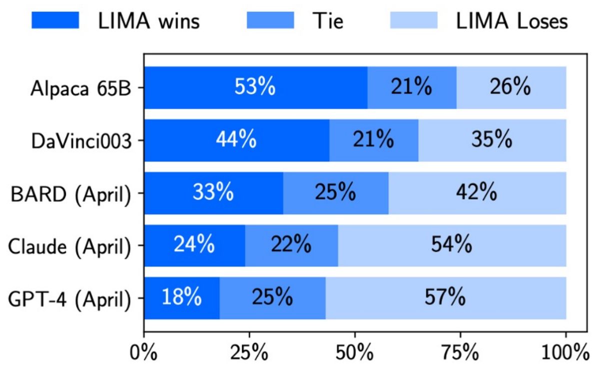


Figure 1: Human preference evaluation, comparing LIMA to 5 different baselines across 300 test prompts.

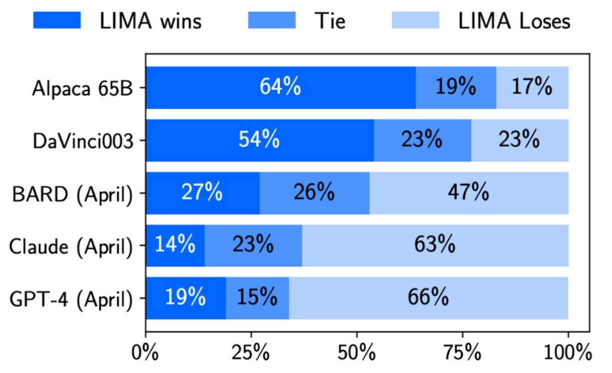
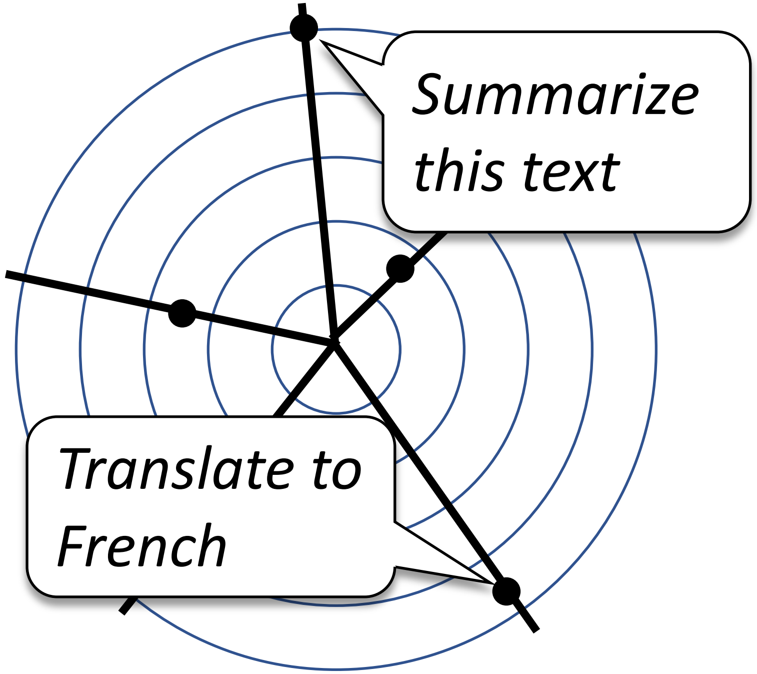
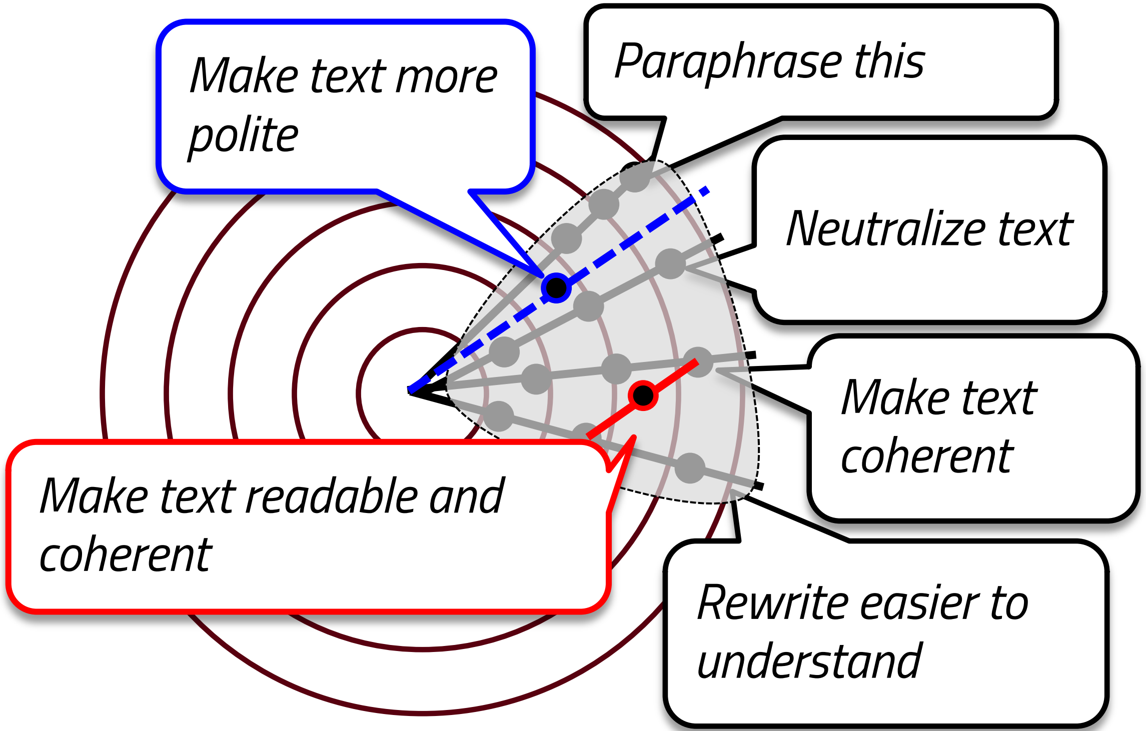


Figure 2: Preference evaluation using GPT-4 as the annotator, given the same instructions provided to humans.

General-purpose (left) vs Task-specific (right) Instruction Tuning



Sparse task distribution



Dense task distribution

Generalize to composite & unseen tasks

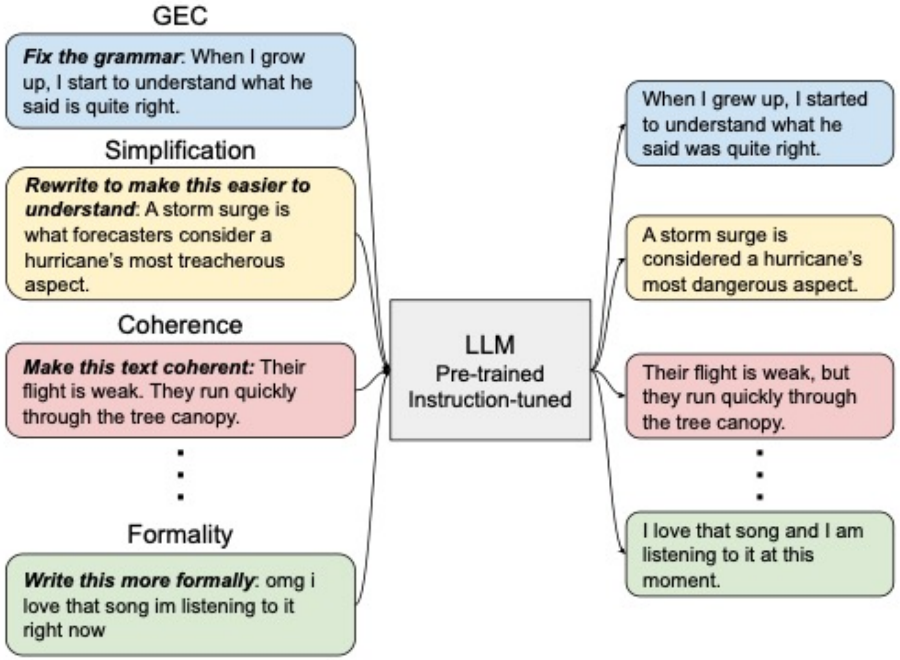
Densifying the task distribution to strictly instructions within the text revision domain can largely improve model performance for revision tasks over scaling model size with general instructions

[Raheja et al., EMNLP Findings 23]

User-driven Revision (CoEdIT)

Conversational Text Editing via Instruction Tuning

Edit Intention / Task	Verbalizers
GEC	Fix grammar, Fix grammar in this sentence, Fix grammar in the sentence, Fix grammar errors, Fix grammatical errors, Fix grammaticality, Fix all grammatical errors, Fix grammatical errors in this sentence, Fix grammar errors in this sentence, Fix grammatical mistakes in this sentence, Fix grammaticality in this sentence, Fix grammaticality of the sentence, Fix disfluencies in the sentence, Make the sentence grammatical, Make the sentence fluent, Fix errors in this text, Update to remove grammar errors, Remove all grammatical errors from this text, Improve the grammar of this text, Improve the grammaticality, Improve the grammaticality of this text, Improve the grammaticality of this sentence, Grammar improvements, Remove grammar mistakes, Remove grammatical mistakes, Fix the grammar mistakes, Fix grammatical mistakes

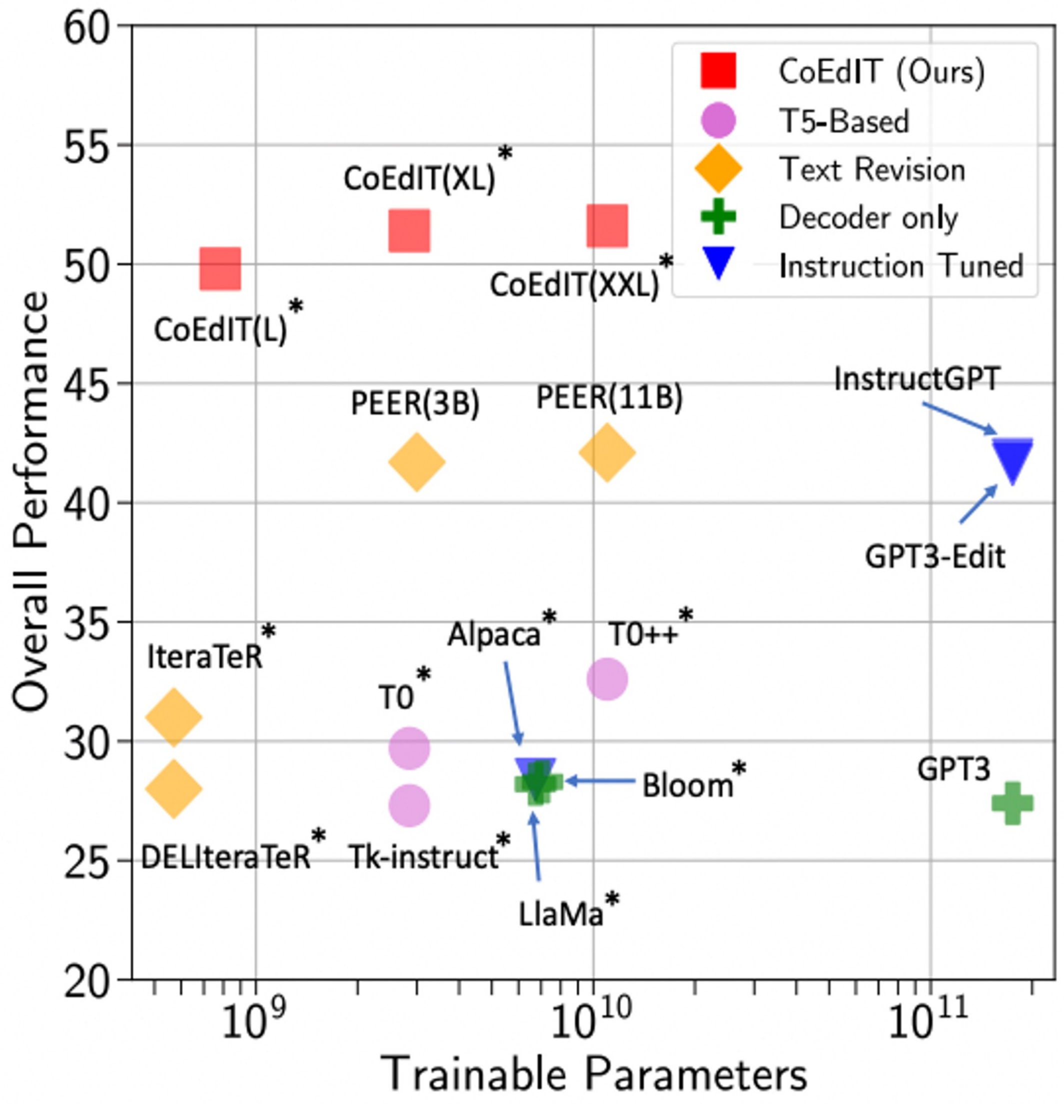


Edit Intention	Datasets	Size	Example Input	Example Output
FLUENCY	NUCLE-14 Lang-8 BEA-19	20k	*Fix the grammar:* When I grow up, I start to understand what he said is quite right.	When I grew up, I started to understand what he said was quite right.
COHERENCE	DiscoFuse	11k	*Make this text coherent:* Their flight is weak. They run quickly through the tree canopy.	Their flight is weak, but they run quickly through the tree canopy.
CLARITY (Simplification)	NEWSELA WikiLarge WikiAuto ParabankV2 ITERATER-CLARITY	13k	*Rewrite to make this easier to understand:* A storm surge is what forecasters consider a hurricane's most treacherous aspect.	A storm surge is considered a hurricane's most dangerous aspect.
STYLE (Paraphrase)	ParabankV2	15k	*Paraphrase this:* Do you know where I was born?	Do you know my birthplace?
STYLE (Formalize)	GYAFC	12k	*Write this more formally:* omg i love that song im listening to it right now	I love that song and I am listening to it at this moment.
STYLE (Neutralize)	WNC	11k	*Write in a more neutral way:* The authors' exposé on nutrition studies.	The authors' statements on nutrition studies.

Table 1: Example data instances in the COEDIT dataset (82K <instruction: source, target> pairs). Instructional prompts in the inputs are *italicized*.

[Raheja et al., EMNLP Findings 23]

Quantitative Evaluations of Text Revision Models



CoEdIT's task specific instruction tuning largely outperforms most generally instruction-tuned models

Overall, CoEdIT generates better text edits than models that are even 60x larger

[Raheja et al., EMNLP Findings 23]

CoEdit on HuggingFace

Model	Number of parameters
CoEdIT-large	770M
CoEdIT-xl	3B
CoEdIT-xxl	11B

https://huggingface.co/grammarly/coedit-large

Downloads last month
29,703

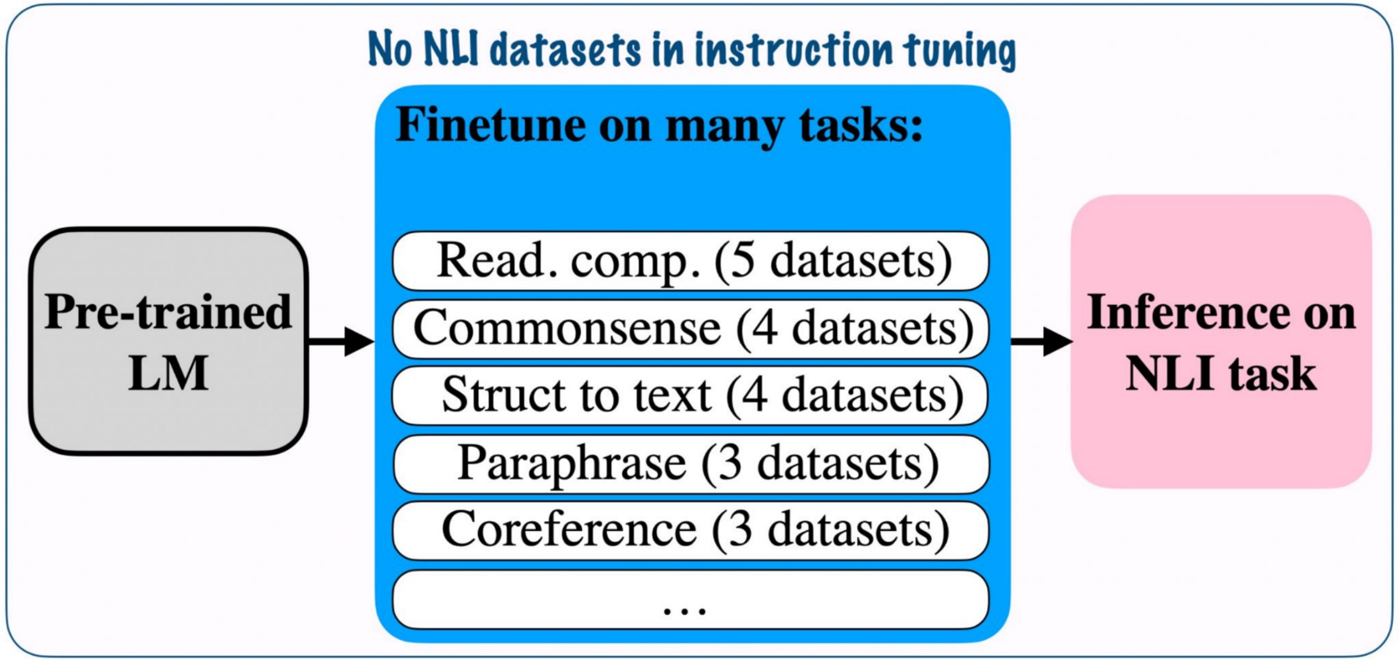
```
from transformers import AutoTokenizer,
T5ForConditionalGeneration

tokenizer =
AutoTokenizer.from_pretrained("grammarly/coedit-large")
model =
T5ForConditionalGeneration.from_pretrained("grammarly/coe
dit-large")
input_text = 'Fix grammatical errors in this sentence:
When I grow up, I start to understand what he said is
quite right.'
input_ids = tokenizer(input_text,
return_tensors="pt").input_ids
outputs = model.generate(input_ids, max_length=256)
edited_text = tokenizer.decode(outputs[0],
skip_special_tokens=True)
```

[Raheja et al., EMNLP Findings 23]

Evaluation

Evaluating on unseen tasks

- We evaluate on "unseen" or "zero-shot" tasks where no datasets from that task were seen during instruction tuning.
  → **Emergent abilities of LLMs!**

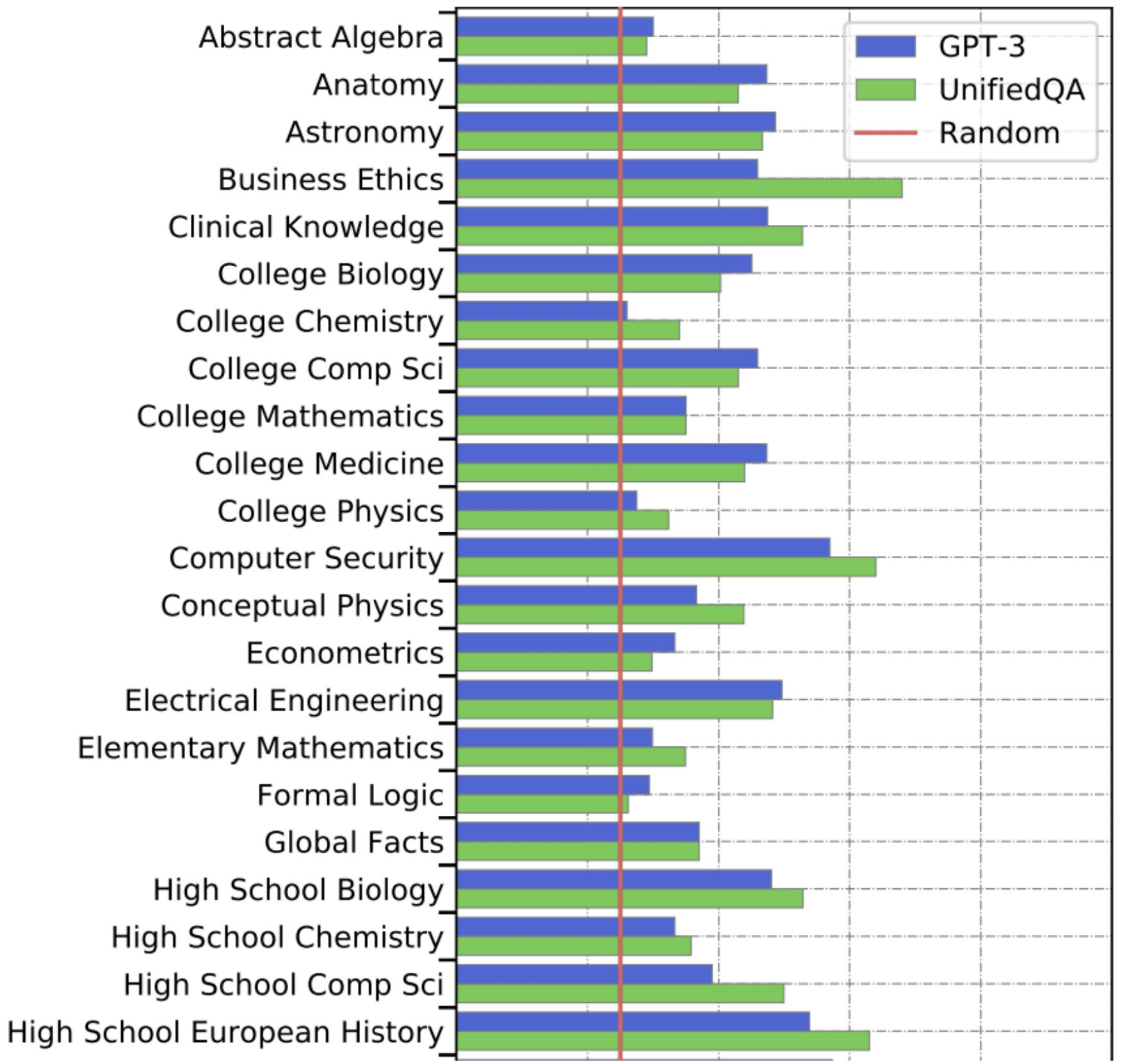


[FLAN-T5; Chung et al., 2022]

Evaluating on new benchmarks

- **Massive Multitask Language Understanding (MMLU)** [Hendrycks et al., 2021]

  New benchmarks consisting of 57 diverse *knowledge intensive* tasks

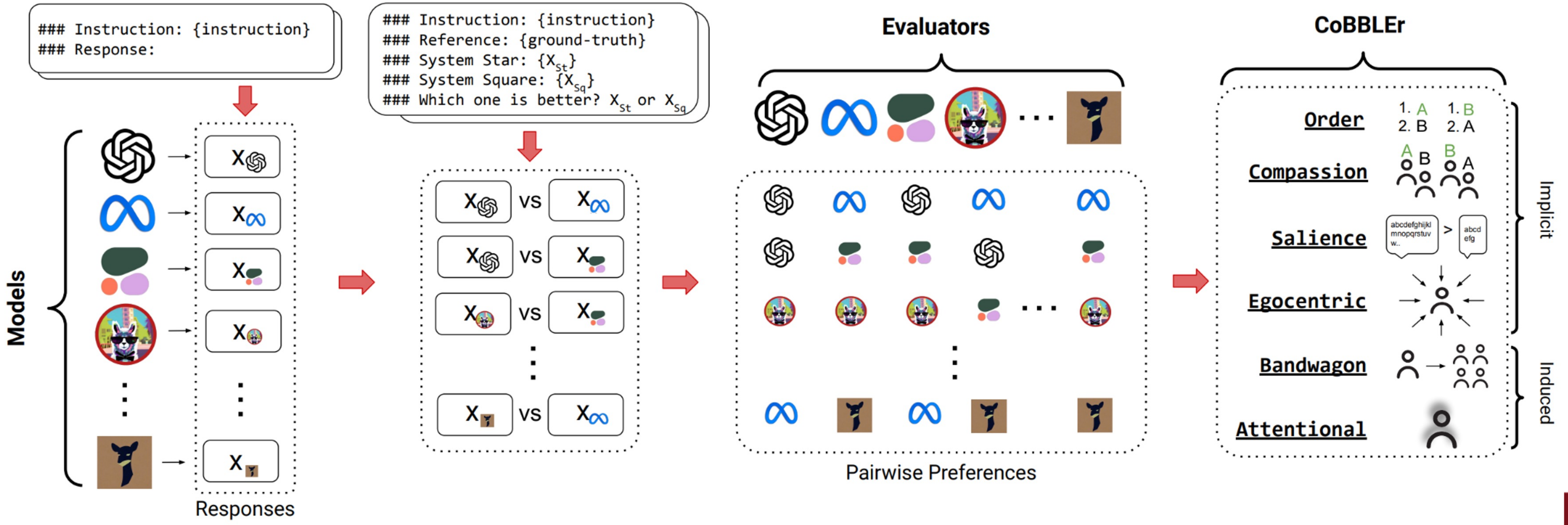


Evaluating on new benchmarks

- **BIG-Bench** [[Srivastava et al., 2022]] with more than 200+ tasks

Evaluating on new benchmarks

- **CoBBLEr** [Koo et al. 2023], *"Benchmarking Cognitive Biases in Large Language Models as Evaluators"*
  - For evaluating LLMs for their capabilities as ***unbiased automatic evaluators***



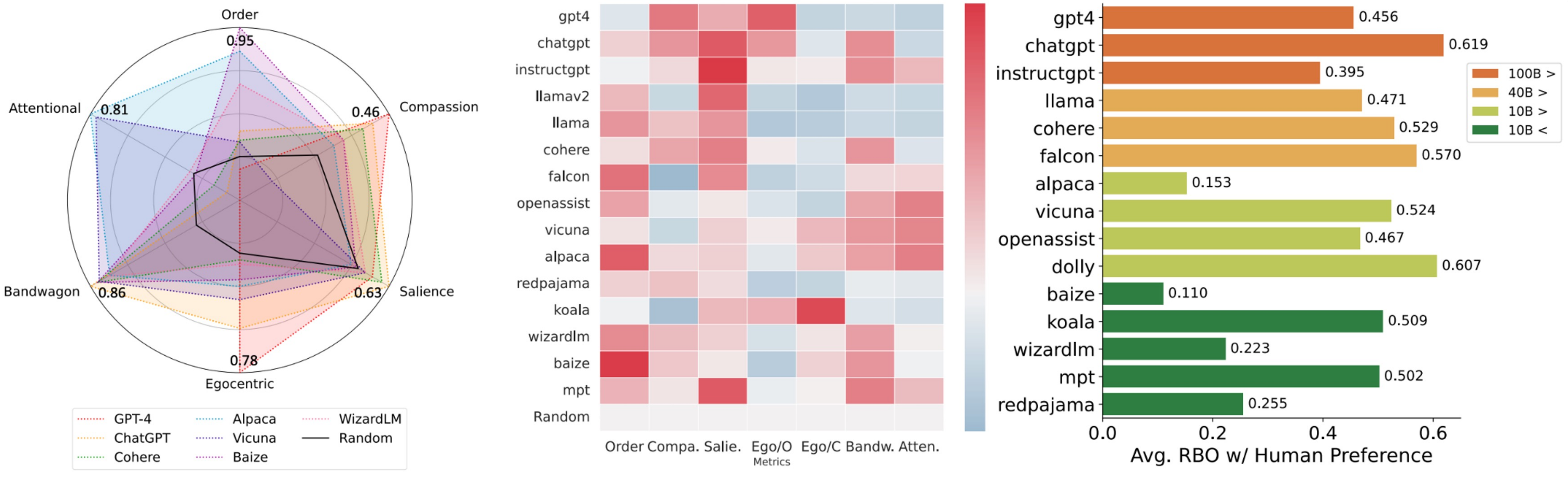
Evaluating on new benchmarks

- **CoBBLEr** [Koo et al. 2023]
  - Implicit biases: general prompt setting
  - Induced biases: try to induce undesired behaviors akin to adversarial attacks

Bias	Bias Behavior	Example
ORDER BIAS	The tendency to give preference to an option based on their order (e.g. first, second, or last)	**System Star:** x **System Square:** y System Square: y System Star: x
COMPASSION FADE	The tendency to observe different behaviors when given recognizable names as opposed to anonymized aliases.	Model Alpaca: x Model Vicuna: y **Model Vicuna:** y **Model Alpaca:** x
EGOCENTRIC BIAS	The inclination to prioritize one's own responses regardless of response quality.	**Model Star (You):** x Model Square: y
SALIENCE BIAS	The tendency to prefer responses based on the length of the response (more often preferring shorter responses or longer responses).	**System Star:** The quick brown fox jumps over the lazy dog. System Square: The fox jumped.
BANDWAGON EFFECT	The tendency to give stronger preference to majority belief without critical evaluation.	**85%** believe that System Star is better.
ATTENTIONAL BIAS	The inclination to give more attention to irrelevant or unimportant details.	System Square likes to eat oranges and apples

Evaluating on new benchmarks

- **CoBBLEr** [Koo et al. 2023]
  - Even popular LLMs like GPT-4 are shown to have biases!
  - So be careful when using them as evaluators in replacement of human workers.



(a) Proportion of biased evaluations (b) Heatmap of bias intensity (c) Correlation with human judgment

Augmented Language Models

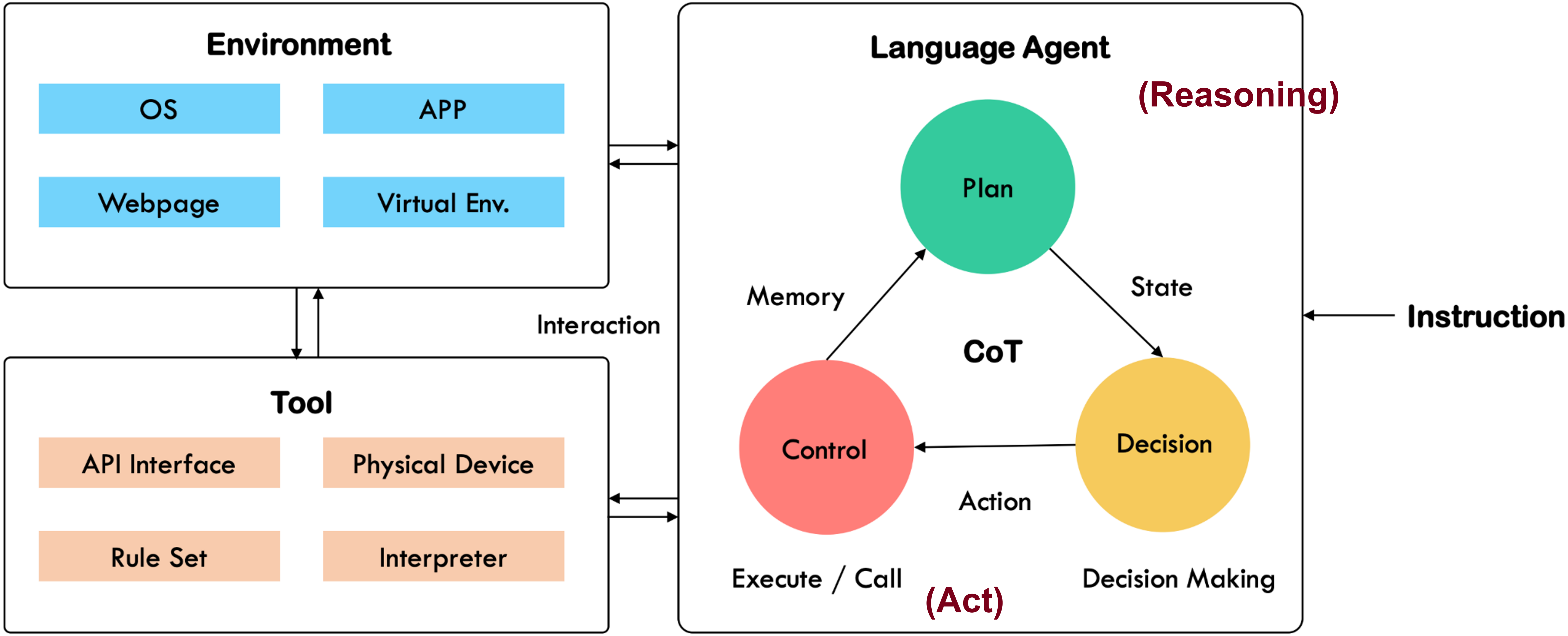
Weakness of LLMs

- Pre-trained LLMs struggle at completing tasks that require:
  - Latest knowledge after the model pretraining time cutoff or
  - Knowledge with internal/private knowledge base
  - Symbolic or other deterministic execution capabilities
- These issues stem from their fundamental limitations:
  - They are trained to perform statistical modeling given a single parametric model and a limited context
  - Their main objective function, *the next token prediction task*, does not cater for explicit symbolic capabilities

Augmented Language Models

- Recent trend is to move slightly away from the purely statistical language modeling and *integrate external components*
  - So that a *more relevant context* is produced at the cost of more computation
  - Resulting in non-parametric models
- An augmentation can be viewed in three dimensions: [Mialon et al. 2023]
  - **Reasoning**: breaking up a complex task into smaller subtasks
  - **Tool**: external modules that can be called
  - **Act**: Calling of a tool to have an effect

Augmented Language Models



[Zhang et al. 2023]

Reasoning

Reasoning

- Reasoning is the ability to make inferences using evidence and logic.
  - Commonsense, mathematical, symbolic, etc.
  - Often this involves deductions from inference chains, i.e. “multi-step reasoning”
- Main challenge is to break down a complex problem into smaller subproblems and generate the solution by composing the (correctly predicted) answers to the subproblems.
- Eliciting reasoning in LLMs
  - Eliciting reasoning with prompting
  - Divide and concur with recursive prompting
  - Teaching LLMs to reason

Eliciting reason with prompting

- Essentially methods can be categorized as either zero-shot or few-shot
- Zero-shot prompting
  - “Let’s think step by step.” [Kojima et al. 2022]
  - “Chain-of-thought decoding” [Wang and Zhou 2024]

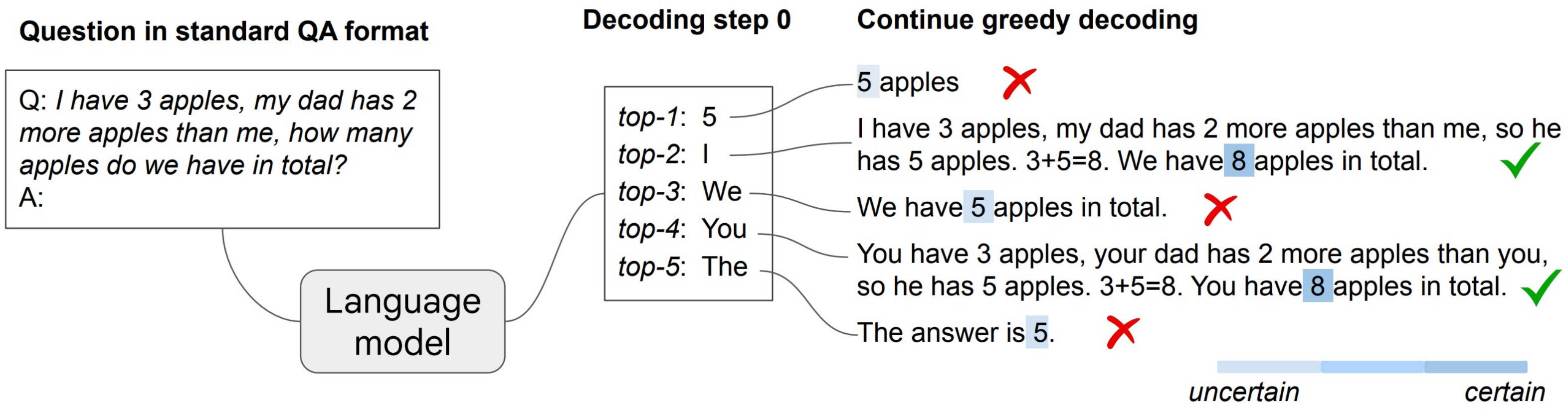
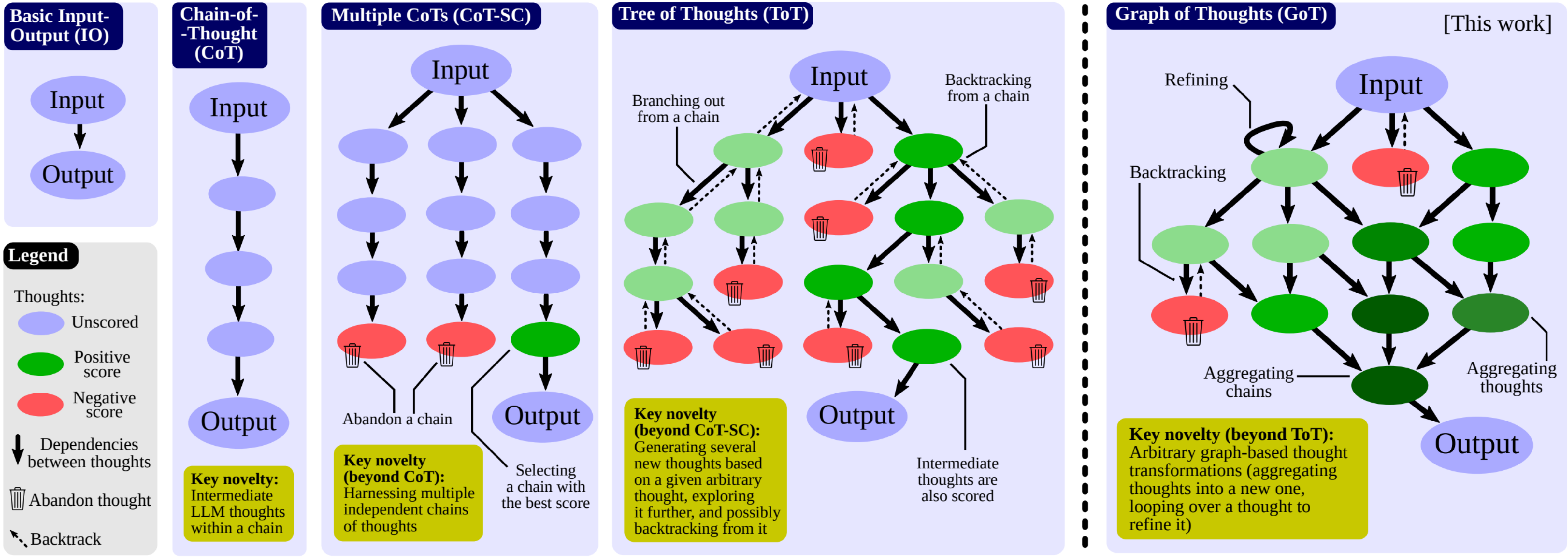


Figure 1 | **Illustration of CoT-decoding.** Pre-trained LLMs are capable of inherent reasoning without prompting by considering alternative top-k tokens, rather than solely relying on the top-1 greedy decoding path. Moreover, these models tend to display higher confidence in decoding the final answer (indicated by a darker shaded color) when a CoT reasoning path is present.

Eliciting reason with prompting

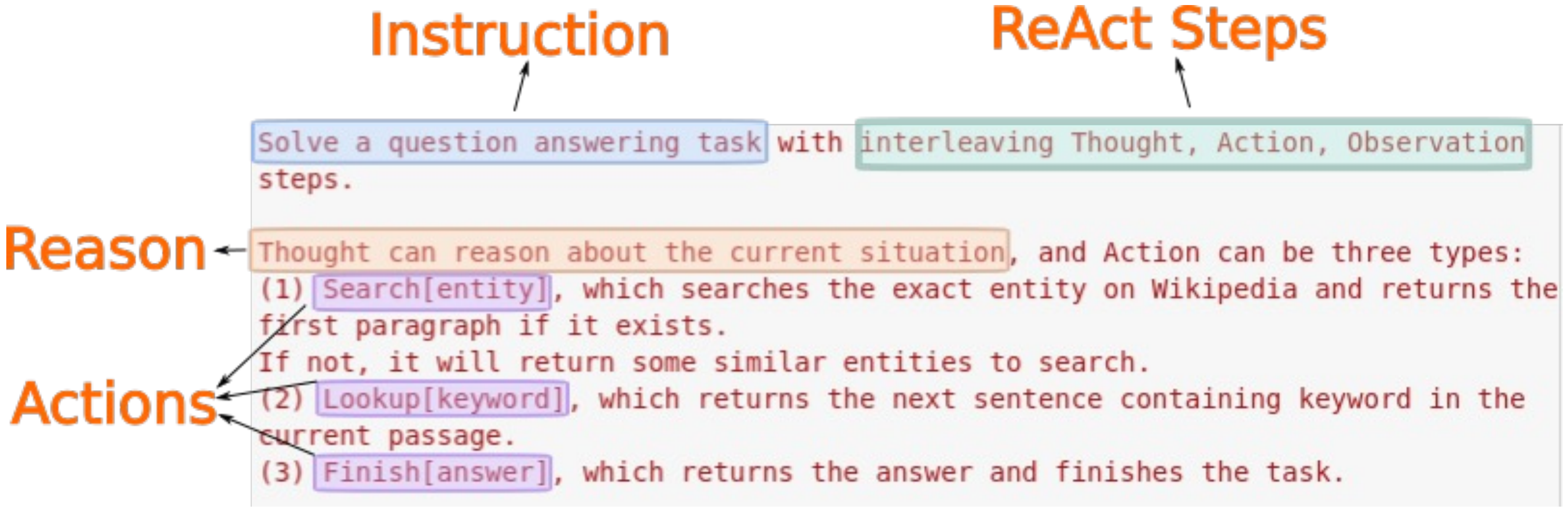
- Few-shot prompting – variants of CoT prompting

[Besta et al. 2024]



Eliciting reason with prompting

- Few-shot prompting – via “programming”
  - ReAct prompting [Yao et al. 2022]



https://www.width.ai/post/react-prompting

Eliciting reason with prompting

- Few-shot prompting – via "programming"
  - ReAct prompting [Yao et al. 2022]

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys]

Programing LLMs for *multi-step reasoning*

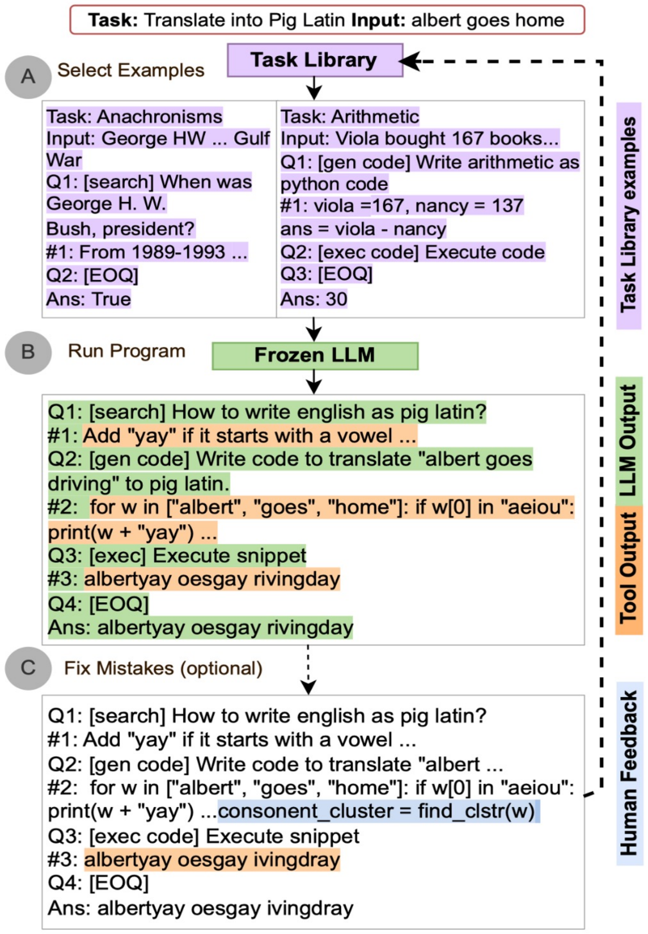


Figure 1: ART generates automatic multi-step decompositions for new tasks by selecting decompositions of related tasks in the *task libray* (A) and selecting and using tools in the *tool library* alongside LLM generation (B). Humans can optionally edit decompositions (eg. correcting and editing code) to improve performance (C).

- **ART** (Automatic multi-step reasoning and tool-use for large language models; Paranjape et al. 2023):
  - ART automatically creates decompositions (multistep reasoning) for examples of new tasks.
  - ART retrieves comparable task instances from a library, enabling quick task analysis and tool application.
  - Using a structured query language, it facilitates reading intermediate stages, pausing for external tool use, and restarting after tool output integration.

    At each step, the framework selects and utilizes the most appropriate tools.

Programing LLMs for *multi-step reasoning*

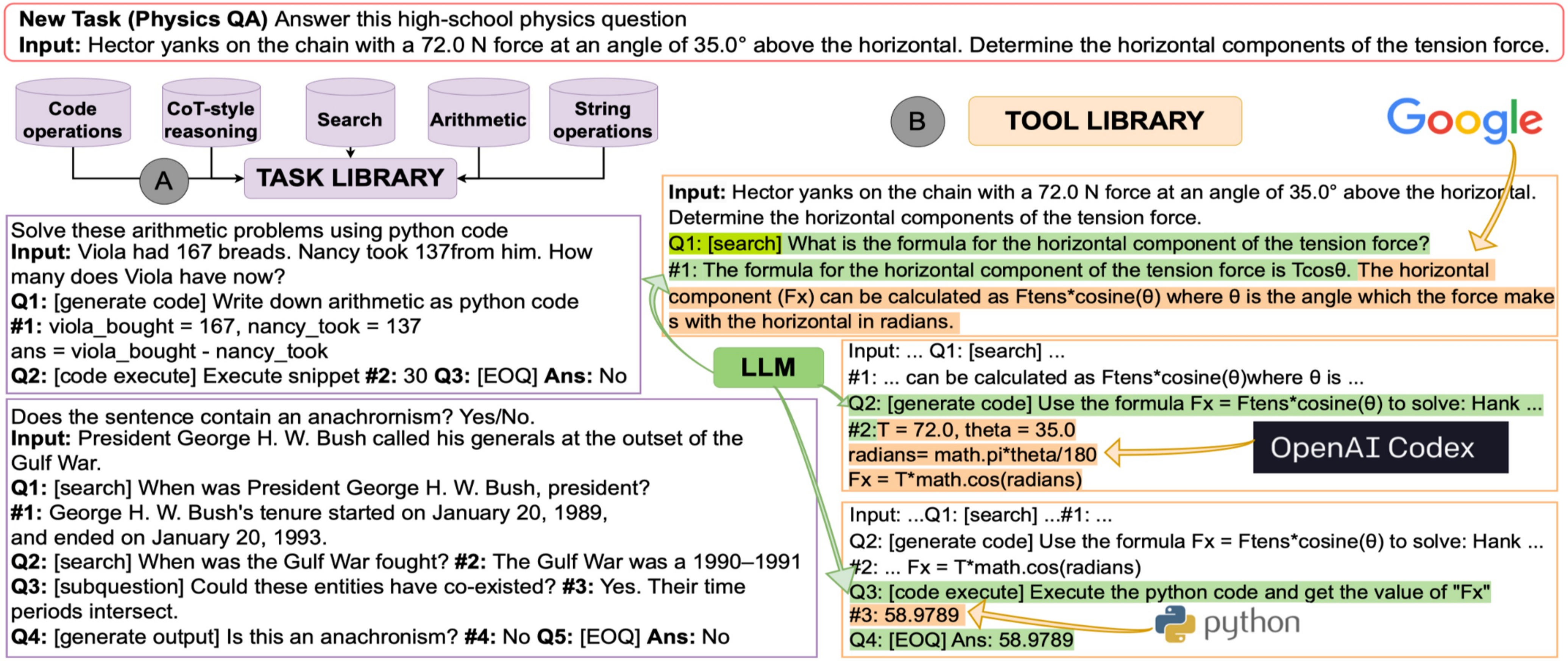


Figure 2: A run-through of ART on a new task, Physics QA. (A) Programs of related tasks like anachronisms and Math QA provide few-shot supervision to the LLM — related sub-steps and tools in these programs can be used by the LLM for cross-task generalization (shown in purple). (B) Tool use: Search is used to find the appropriate physics formula, and code generation and execution are used to substitute given values and compute the answer (shown in orange).

Programing LLMs for *multi-step reasoning*

Task Name (Cluster)	Few Shot	AutoCot	ART w/o Tool Use	ART	GPT-3 Best
Test Tasks					
Sentence Ambiguity (Search)	70.67[5]	51.47	71.00	73.33	-
Strategy QA (Search)	55.49[5]	27.22	59.37	66.44	-
Physics (Search)	70.09[5]	61.83	59.13	67.55	-
Δ with ART (Search)	**+3.7**	**+22.27**	**+ 5.9**		
Physics Questions (Arithmetic)	7.02[5]	5.56	6.30	20.37	-
Operators (Arithmetic)	71.23[7]	75.52	71.80	92.00	-
Unit interpretation (Arithmetic)	58.2[7]	41.20	51.4	53.99	-
Repeat copy logic (Arithmetic)	50.01[7]	15.63	31.25	44.38	-
Object Counting (Arithmetic)	39.2[7]	26.80	42.2	87.00	81.20[1]
Penguins in a table (Arithmetic)	58.23[7]	40.40	68.86	77.85	72.34[1]
Reasoning about objects (Arithmetic)	71.00[7]	33.33	45.35	64.34	52.69[1]
Tracking shuffled objects (Arithmetic)	22.39[7]	19.44	18.14	37.67	36.32[1]
Δ with ART (Arithmetic)	**+19.0**	**+36.7**	**+ 23.1**		**+6.1**
Word Unscramble (String)	40.72[7]	32.44	23.03	42.7	-
Simple Text Editing (Code)	35.31[5]	30.21	20.74	27.65	-
CS Algorithms (Code)	73.48[7]	0.0	41.59	88.11	-
Sports Understanding (CoT)	69.74[5]	51.47	92.89	-	86.59[1]
Snarks (CoT)	54.58[5]	57.24	57.13	-	65.2[1]
Disambiguation QA (Free-form)	55.03[5]	48.45	55.89	-	60.62[1]
Temporal sequences (CoT)	55.80[7]	19.70	49.5	-	81.8[1]
Ruin names (CoT)	71.01[5]	55.28	60.22	-	-
Δ with ART (Misc)	**2.4**	**22.5**	**24.37**		-9.4
Δ with ART (Overall)	**+6.9**	**+24.6**	**+16.7**		-1.7
MMLU					
College Computer Science (Search)	41.00	43.99	63.40	67.80	63.6[6]
Astronomy (Search)	62.10	41.48	76.71	79.1	62.5[6]
Business Ethics (Search)	61.60	48.8	77.17	81.16	72.7[6]
Virology (Search)	50.03	49.52	71.60	71.49	50.72[6]
Geography (Search)	77.67	57.07	70.30	71.71	81.8[6]
Mathematics (Arithmetic)	36.67	33.77	39.50	45.66	34.5[6]
Δ with ART (MMLU)	**+14.6**	**+23.7**	**+3.0**		**+8.5**

Table 3: ART performance on BigBench tasks and MMLU tasks. ([1] Human-crafted CoT (Wei et al., 2022; Suzgun et al., 2022), [5] InstructGPT (Ouyang et al., 2022), [6] Scaled instruction finetuning (Chung et al., 2022), [7] Code-davinci-002 (Chen et al., 2021)).

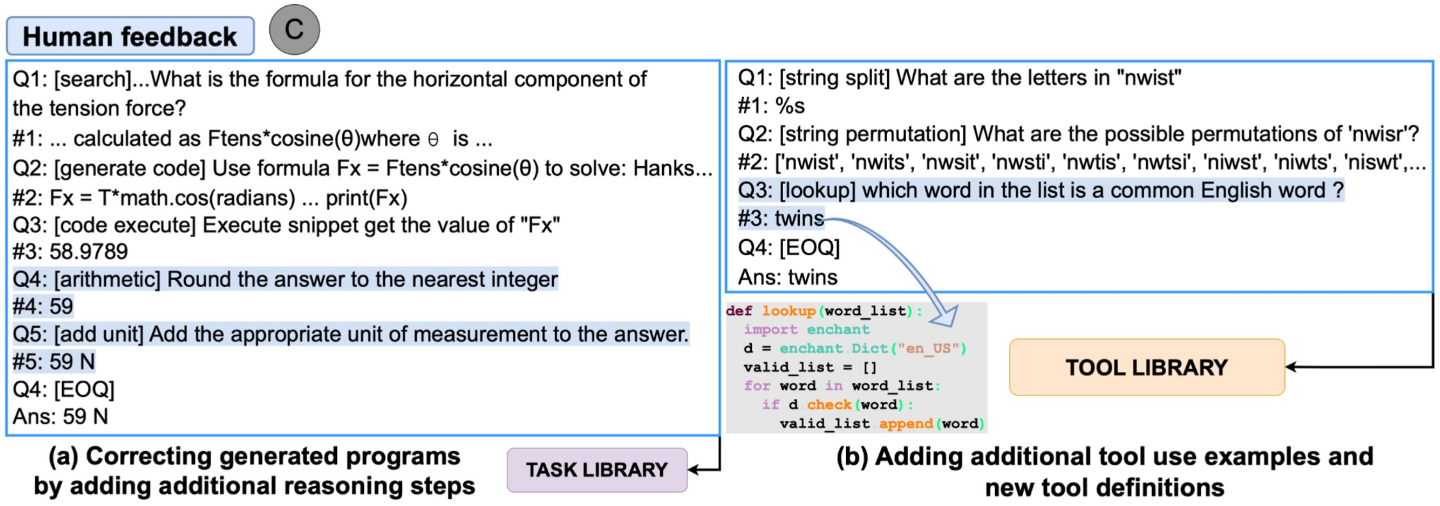


Figure 3: Human feedback to ART shown for (a) PQA where reasoning steps are added to the program and; (b) Word unscrambling where tool library is augmented with a new lookup tool.

Using Tools and Act

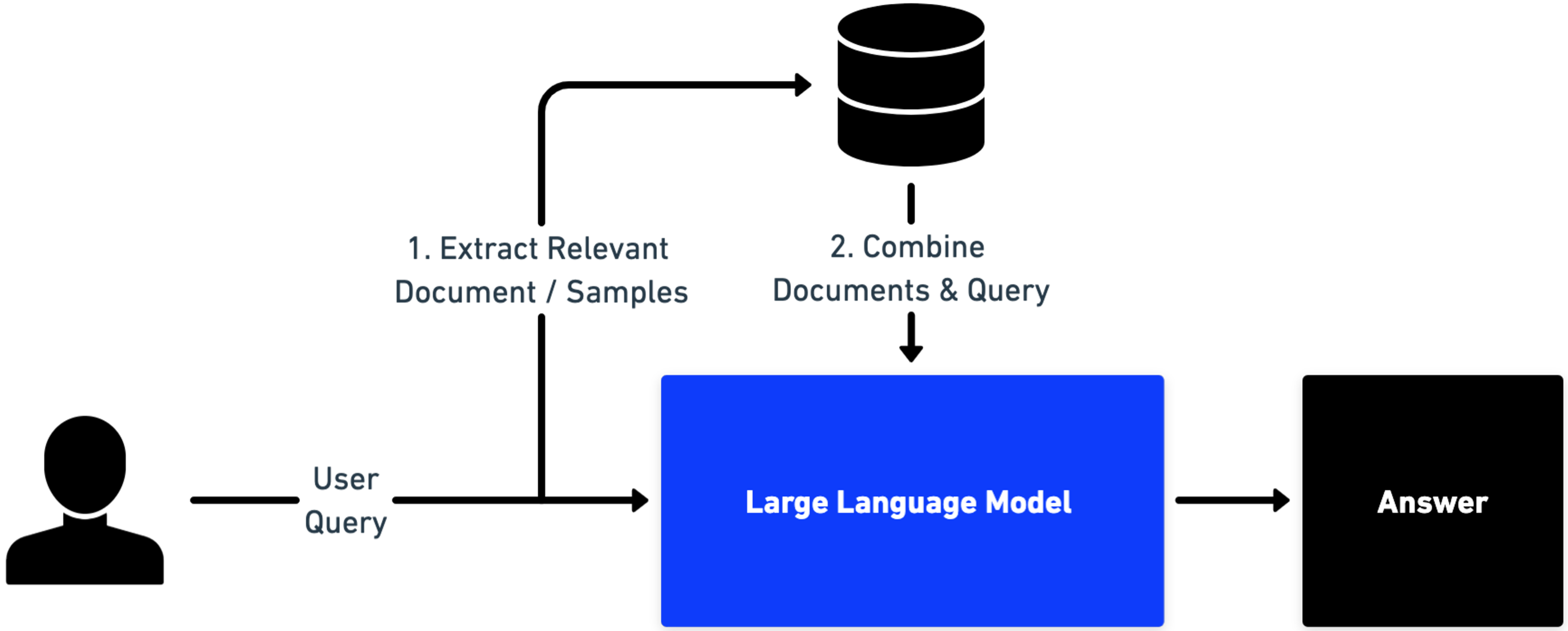
Tools and Act

- Expanding LLMs' capabilities to access and leverage external knowledge and computational resources beyond what is stored in their own weights.
- This integration, via special tokens, allows LLMs to perform tasks like:
  - Exact computation
  - Symbolic reasoning
  - Information retrieval, etc.
- Tools can range from:
  - Calling (another) LLM or other programs like Python interpreter
  - Querying (vector) databases, knowledge graphs, or search engines
    - Often referred to as “information retrieval”

Tools and Act

- Use Google Search for document retrieval to augment LLMs. [Lazaridou et al. 2022]
  - Given a question, clean text (paragraph) is extracted out of 20 URLs returned by Google.
  - Paragraphs are ranked by TF-IDF based cosine similarity between evidence paragraphs and the query.
  - Only the most relevant paragraph is used in the prompt to produce an answer

Augmenting LLMs with Retrieval



Augmenting LLMs with Retrieval

- “Internal retrieval” from LLMs could be also beneficial
  - Generate knowledge about a topic before answering the question [Liu et al. 2022]

```
Generate some knowledge about the input. Examples:

Input: What type of water formation is formed by clouds?
Knowledge: Clouds are made of water vapor.

Input: {question}
Knowledge:
```

Augmenting LLMs

- "Internal retrieval" from LLMs could be also beneficial
  - Iterative prompting of LLM for text refinement task [Schick et al. 2022]
  - The "plan" can come from human users too.

Iteration 0

Text: Brittney Reese (born September 9, 1986 in Gulfport, Mississippi) is an American long jumper.
<LM>
Plan: Remove incorrect information
Edit: Brittney Reese (born September 9, 1986 ~~in Gulfport, Mississippi~~) is an American long jumper.
</LM>

Iteration 1

Text: Brittney Reese (born September 9, 1986) is an American long jumper.
<LM>
Plan: Add information about her career
Edit: Brittney Reese (born September 9, 1986) is an American long jumper , who competed at the 2008 Summer Olympics, and is a 4-time World Champion .
</LM>

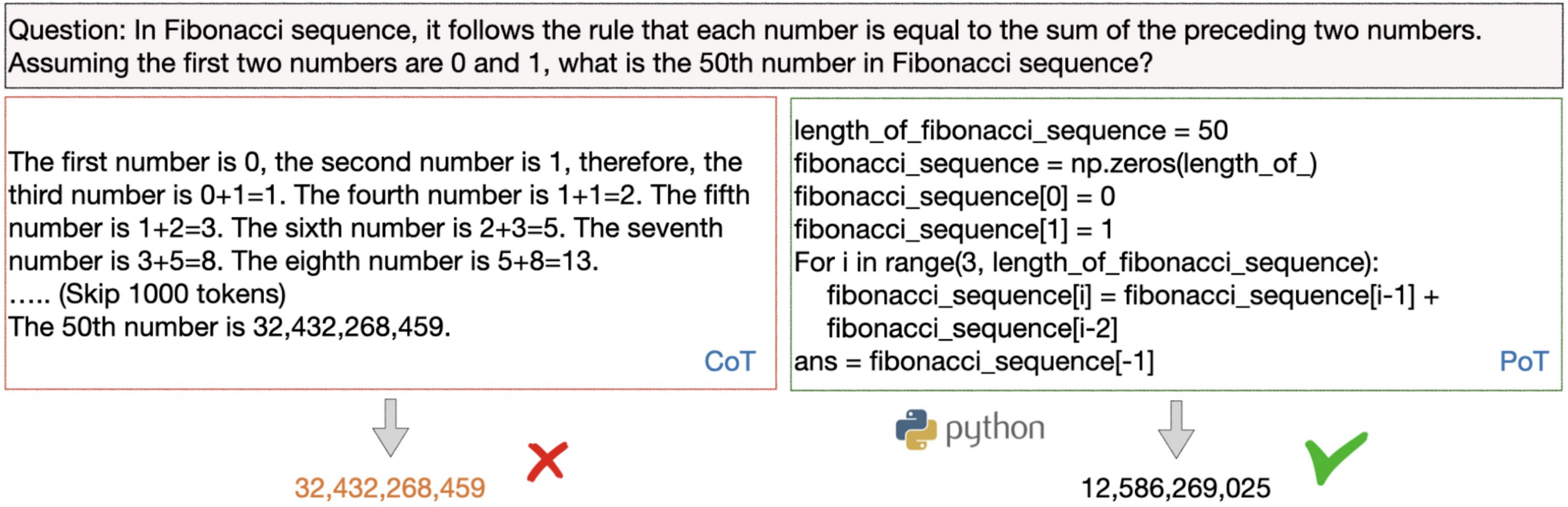
Iteration 2

Text: Brittney Reese (born September 9, 1986) is an American long jumper, who competed at the 2008 Summer Olympics, and is a 4-time World Champion.
<LM>
Plan: Add her birthplace
Edit: Brittney Reese (born September 9, 1986 in Inglewood, California) is an American long jumper, who competed at the 2008 Summer Olympics, and is a 4-time World Champion.
</LM>

Figure 5: Iterative prompting example using PEER (Schick et al., 2022), a LM trained to produce a plan of action and edit to the input text at each step. This process can be repeated until the generated text requires no further updates. <LM> denotes the start of the LM's output to the prompt, while </LM> denotes the end.

Augmenting LLMs with Programming Language

- Ask LLM to generate programming language statements to resolve natural language reasoning problems, hence offloading the solution step to a runtime such as a Python interpreter.



PAL (Program-aided language models); Gao et al. 2022 and **PoT** (Program of Thoughts prompting); Chen et al. 2022

Augmenting LLMs with External APIs

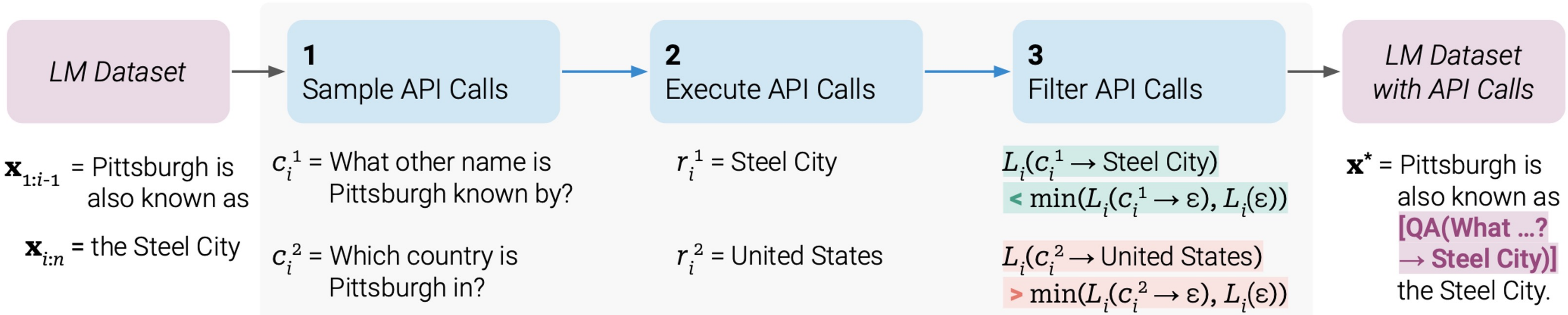
- **TALM** (Tool Augmented Language Models; Parisi et al. 2022): LM augmented with text-to-text API calls.
  - LM is guided to generate |tool-call and tool input text conditioned on task input text to construct API call requests
  - When |result shows up, the specified tool API is called and the returned result gets appended to the text sequence as |output token.

An abstract task:
task input text |***tool-call*** tool input text |***result*** tool output text |***output*** task output text

A weather task:
how hot will it get in NYC today? |***weather*** lookup region=NYC |***result*** precipitation chance: 10, high temp: 20c, low-temp: 12c |***output*** today's high will be 20C

Augmenting LLMs with External APIs

- **Toolformer** ([Schick et al. 2023](#)) use external tools via simple APIs, which is built in a *self-supervised manner* and only requires a handful of demonstrations for each API.
  - **Calculator** to help LM with the lack of precise math skills;
  - **Q&A** system to help with unfaithful content and hallucination;
  - **Search engine** to provide up-to-date information after pretraining cut off time;
  - **Translation system** to improve performance on low resource language;
  - **Calendar** to make LM be aware of time progression.



Toolformer Training

Step 1:

Prompting to annotate potential API calls.

Ask a pre-trained LM to annotate a dataset via few-shot learning with API call usage examples.

The New England Journal of Medicine is a registered trademark of **[QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society]** the MMS.

Out of 1400 participants, 400 (or **[Calculator(400 / 1400) → 0.29]** 29%) passed the test.

The name derives from "la tortuga", the Spanish word for **[MT("tortuga") → turtle]** turtle.

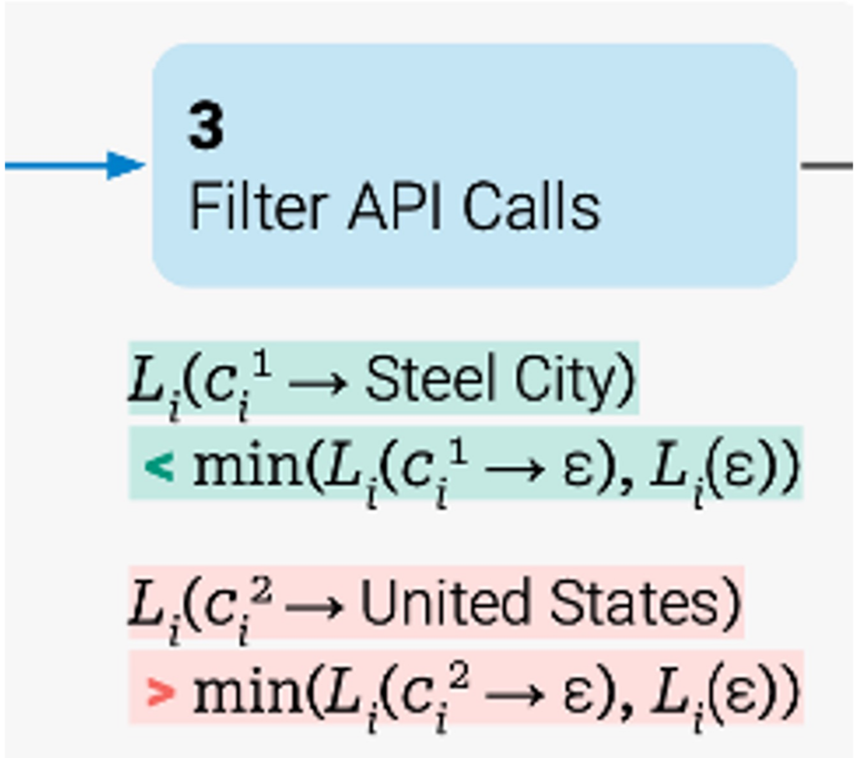
The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their meetings open to the public.

Toolformer Training

Step 2:

Filter annotations based on whether API calls help predict future tokens.

Use a self-supervised loss to decide which API calls are actually helpful.



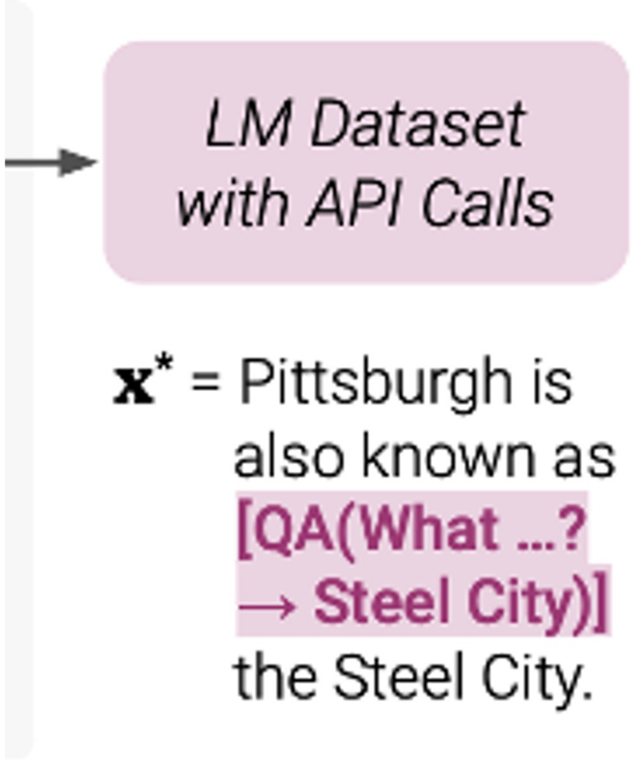
$$L_i^+ = L_i(e(c_i, r_i))$$
$$L_i^- = \min(L_i(\varepsilon), L_i(e(c_i, \varepsilon)))$$

Toolformer Training

Step 3:

Fine-tune LM on this annotated dataset.

The training data is a combination of the original dataset and its augmented version.



Programing LLMs for *controlled generation*

- **Guidance** (Microsoft AI):
  - Allows users to ***constrain generation*** (e.g. with regex and CFGs) as well as to ***interleave control*** (conditional, loops) and generation seamlessly.

Basic generation

An `lm` object is immutable, so you change it by creating new copies of it. By default, when you append things to `lm`, it creates a copy, e.g.:

```
from guidance import models, gen, select
llama2 = models.LlamaCpp(model)

# llama2 is not modified, `lm` is a copy of `llama2` with 'This is a prompt' appended to its state
lm = llama2 + 'This is a prompt'
```

> This is a prompt

You can append *generation* calls to model objects, e.g.

```
lm = llama2 + 'This is a prompt' + gen(max_tokens=10)
```

> This is a prompt for the 2018 NaNoWr

You can also interleave generation calls with plain text, or control flows:

```
# Note how we set stop tokens
lm = llama2 + 'I like to play with my ' + gen(stop=' ') + ' in' + gen(stop=['\n', '.', '!'])
```

> I like to play with my food. in the kitchen

Programing LLMs for *controlled generation*

Constrained Generation

Select (basic)

`select` constrains generation to a set of options:

```
lm = llama2 + 'I like the color ' + select(['red', 'blue', 'green'])
```

I like the color red

Regex to constrain generation

Unconstrained:

```
lm = llama2 + 'Question: Luke has ten balls. He gives three to his brother.\n'
lm += 'How many balls does he have left?\n'
lm += 'Answer: ' + gen(stop='\n')
```

Question: Luke has ten balls. He gives three to his brother.
How many balls does he have left?
Answer: He has seven balls left.

Constrained by regex:

```
lm = llama2 + 'Question: Luke has ten balls. He gives three to his brother.\n'
lm += 'How many balls does he have left?\n'
lm += 'Answer: ' + gen(regex='\d+')
```

Question: Luke has ten balls. He gives three to his brother.
How many balls does he have left?
Answer: 7

Regex as stopping criterion

Unconstrained:

```
lm = llama2 + '19, 18,' + gen(max_tokens=50)
```

19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4,

Stop with traditional stop text, whenever the model generates the number 7:

```
lm = llama2 + '19, 18,' + gen(max_tokens=50, stop='7')
```

19, 18, 1

Programing LLMs for *controlled generation*

- Easy tool use: where the model stops generation when a tool is called, calls the tool, then resumes generation.

```python
@guidance
def add(lm, input1, input2):
    lm += f' = {int(input1) + int(input2)}'
    return lm
@guidance
def subtract(lm, input1, input2):
    lm += f' = {int(input1) - int(input2)}'
    return lm
@guidance
def multiply(lm, input1, input2):
    lm += f' = {float(input1) * float(input2)}'
    return lm
@guidance
def divide(lm, input1, input2):
    lm += f' = {float(input1) / float(input2)}'
    return lm
```

Now we call `gen` with these tools as options. Notice how generation is stopped and restarted automatically:

```python
lm = llama2 + '''\
1 + 1 = add(1, 1) = 2
2 - 3 = subtract(2, 3) = -1
'''
lm + gen(max_tokens=15, tools=[add, subtract, multiply, divide])
```

1 + 1 = add(1, 1) = 2
2 - 3 = subtract(2, 3) = -1
3 * 4 = multiply(3, 4) = 12.0
4 / 5 = divide(4, 5) = 0.8

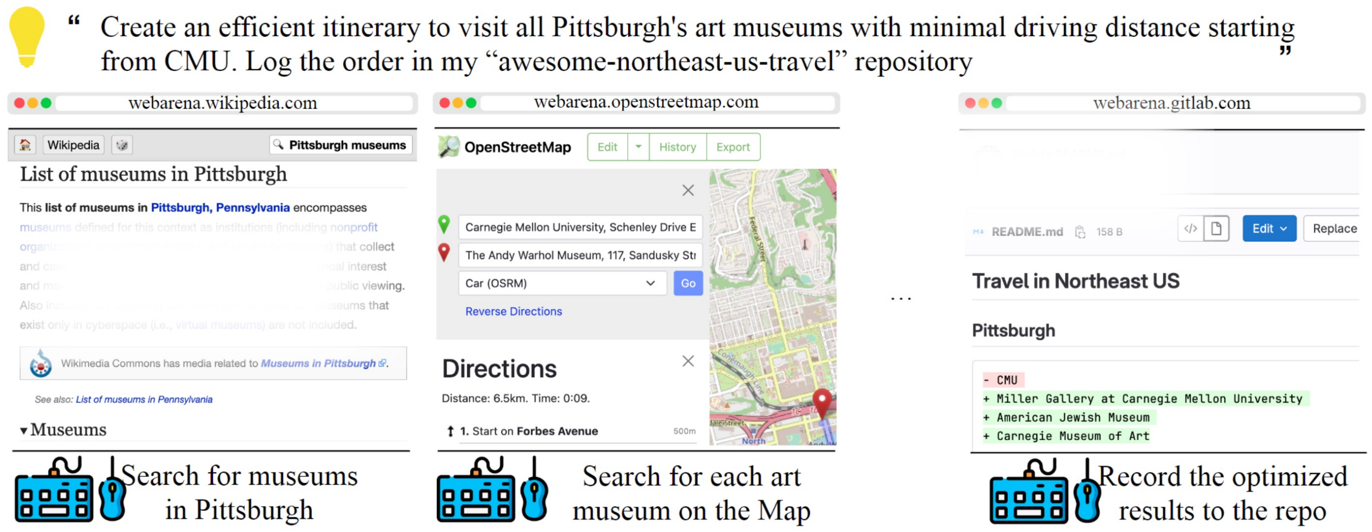
LLMs as Agents

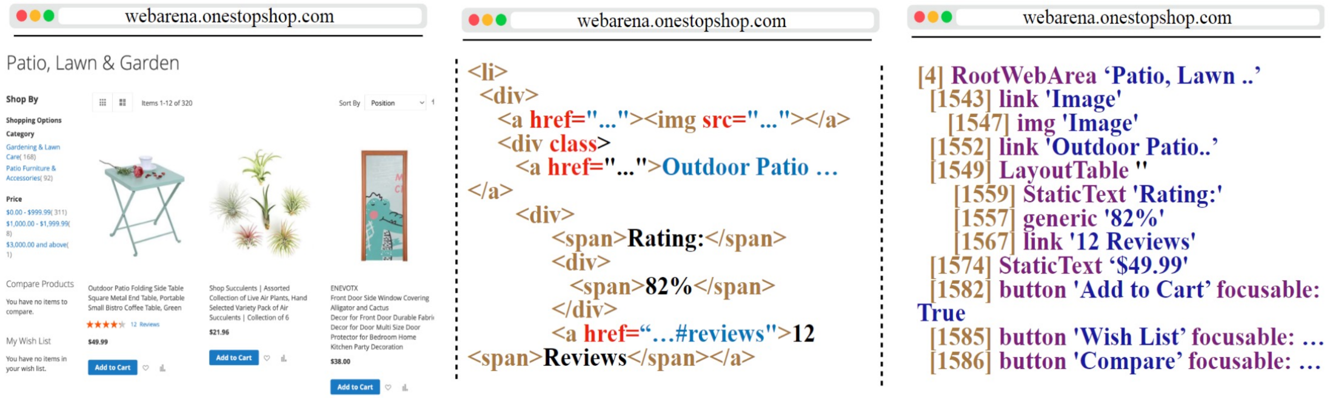
Agents

- Leveraging these internal reasoning capabilities of LLMs and usage of external tools and memory, LLMs can now be considered as agents:
- LLMs as autonomous agents
  - They can be tasked with a realistic and high-level goal
    - "Gather and summarize recent papers published in this month and create a GitHub repository with them"
  - They can break down the goal, execute and evaluate the intermediate steps required to achieve the goal.
- LLMs as social agents
  - They can simulate human interaction
    - Are LLMs' interactions aligned with human values?
    - Can they replace human crowd workers on mundane tasks?

WebArena: a web environment for building autonomous agents [Zhou et al. 2023]

- Simulating an autonomous agent for high-level tasks in e-commerce, social forums, software development, and content management.
- A GPT-4-based agent, show a significant gap between current AI performance (14.41% success rate) and human performance (78.24%)
- https://webarena.dev/





We design the observation to be the URL and the content of a web page, with options to represent the content as a screenshot (left), HTML DOM tree (middle) and accessibility tree (right).

Generative Agents: Interactive Simulacra of Human Behavior [Park et al. 2023]

- Simulating human behavior akin to The Sims
- Agents can:
  - Wake up, cook breakfast, head to work.
  - Notice and converse with each other
  - Remember and reflect
  - And plan the next days

Figure 1: Generative agents are believable simulacra of human behavior for interactive applications. In this work, we demonstrate generative agents by populating a sandbox environment, reminiscent of The Sims, with twenty-five agents. Users can observe and intervene as agents plan their days, share news, form relationships, and coordinate group activities.

Concluding Remarks

- On alignment
  - Training LLMs on various tasks enhances their ability to execute instructions for new tasks.
  - Instruction-tuned LLMs exhibits better zero-shot and few-shot capability.
  - The number of instruction tuning clusters and the scale of the model are important to the performance of LLMs.
  - Many new benchmarks have been developed to test the models capability as well as to alert their pitfalls.
- On augmentation
  - The emergence of reasoning abilities in LLMs facilitates the incorporation of various tools and knowledge bases, thereby greatly enhancing their overall capability.
  - Deciding which tools to use and what relevant external knowledge to integrate is the key.
  - As tasks get more complex, LLMs' long-term planning capability is put to test.