

# CSCE 670 - Information Storage and Retrieval

# Week 11: Neural Recommendation (Cont'd), Large Language Models

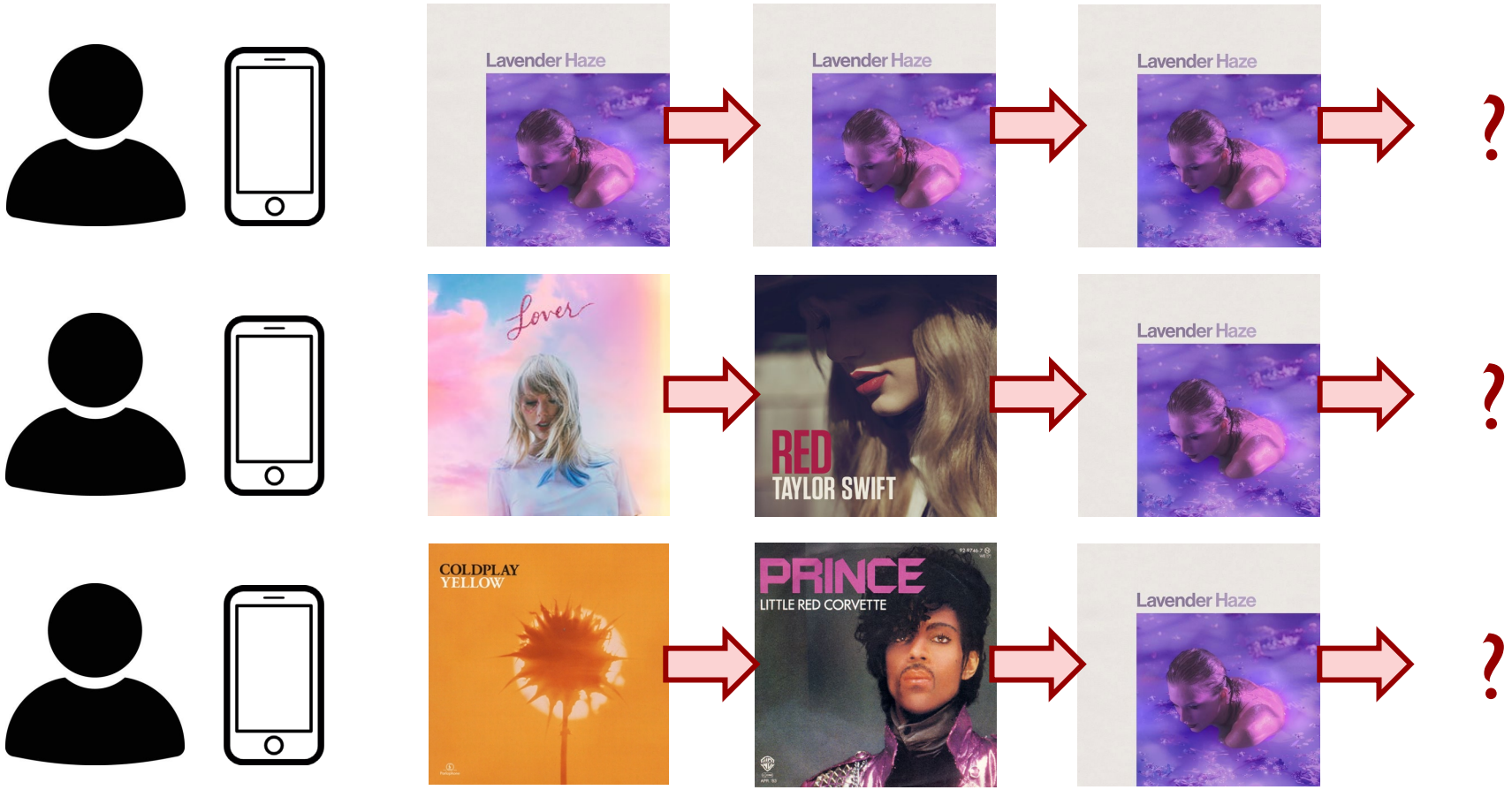
Yu Zhang

yuzhang@tamu.edu

Course Website: https://yuzhang-teaching.github.io/CSCE670-F26.html

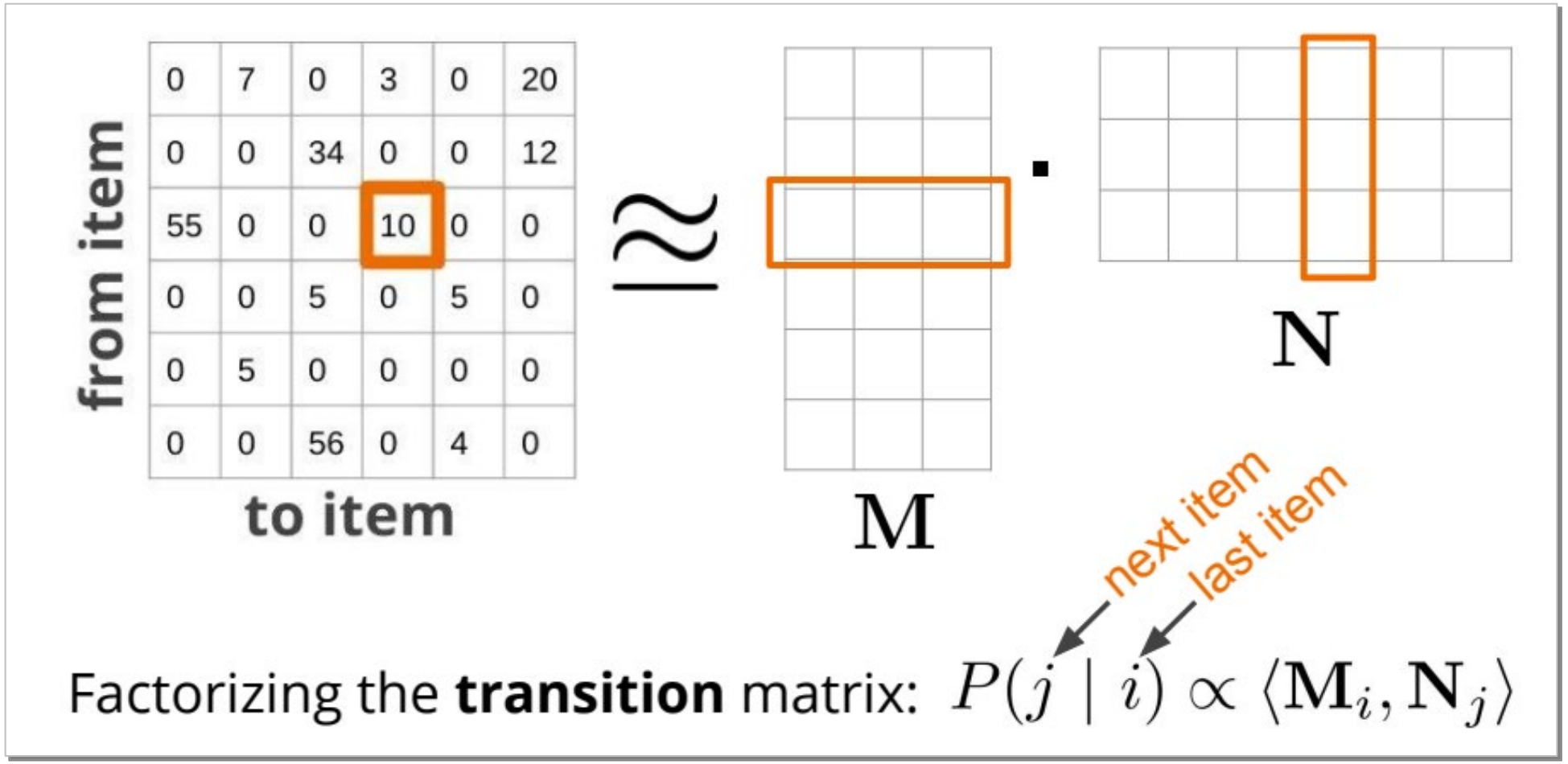
# Recap: Sequential Recommendation

- Understanding dynamic user intent from sequential interactions



# Sequential recommendation is asymmetric!

- One solution: Transition matrix is asymmetric; the Fossil algorithm



Translation is asymmetric!

# TransRec [He et al., RecSys 2017]

## Translation-based Recommendation

Ruining He
UC San Diego
r4he@cs.ucsd.edu

Wang-Cheng Kang
UC San Diego
wckang@eng.ucsd.edu

Julian McAuley
UC San Diego
jmcauley@cs.ucsd.edu

**ABSTRACT**

Modeling the complex interactions between users and items as well as amongst items themselves is at the core of designing successful recommender systems. One classical setting is predicting users' personalized sequential behavior (or 'next-item' recommendation), where the challenges mainly lie in modeling 'third-order' interactions between a user, her previously visited item(s), and the next item to consume. Existing methods typically decompose these higher-order interactions into a combination of *pairwise* relationships, by way of which user preferences (user-item interactions) and sequential patterns (item-item interactions) are captured by separate components. In this paper, we propose a unified method, *TransRec*, to model such third-order relationships for large-scale sequen-

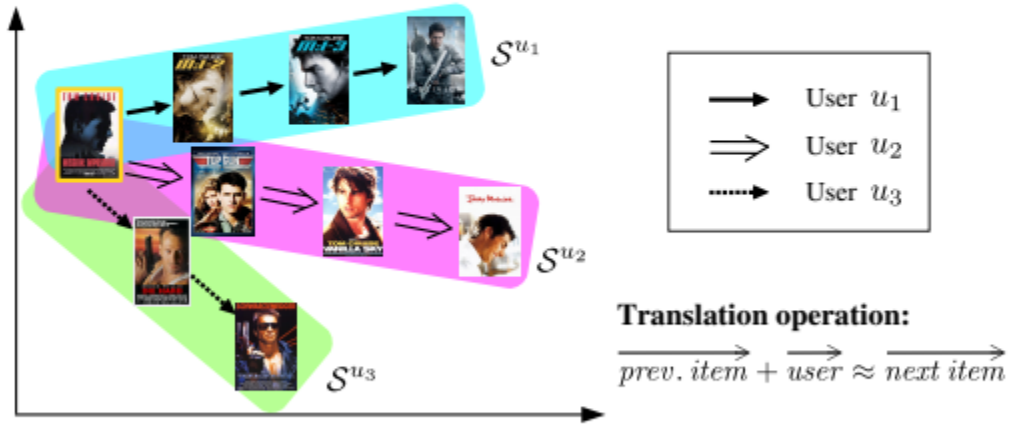
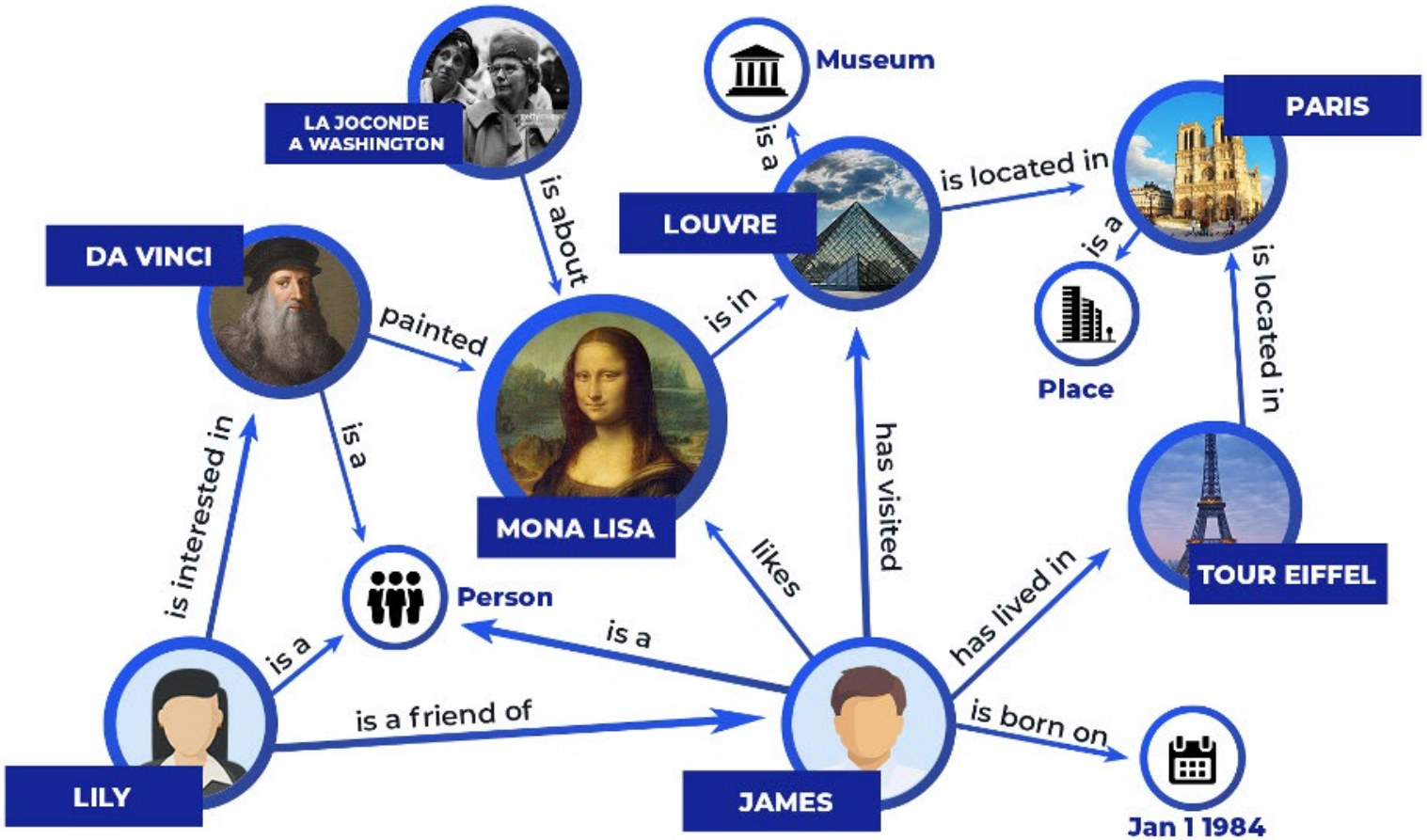
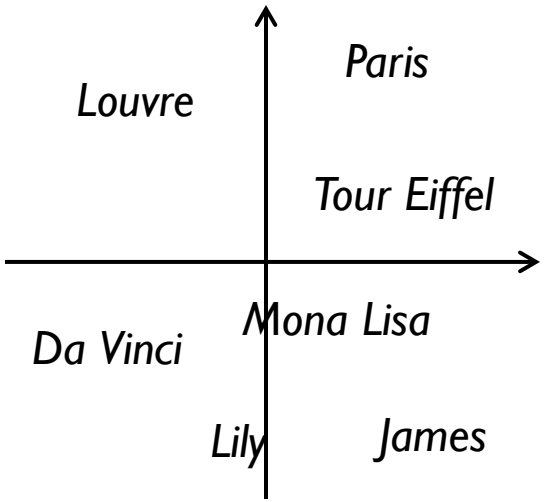



**Figure 1: *TransRec* as a sequential model: Items (movies) are embedded into a 'transition space' where each user is modeled by a *translation* vector. The transition of a user from**

# Background: Translation-Based Embedding

- You have a knowledge graph, and you want to embed all the nodes into a vector space



# Background: Translation-Based Embedding

- Each edge can be represented as a triplet $(h, r, t)$

| head entity $h$ | relation $r$ | tail entity $t$ |
|---|---|---|
| *Tour Eiffel* | is located in | *Paris* |
| *Da Vinci* | painted | *Mona Lisa* |
| *Lily* | is a friend of | *James* |
| … | … | … |

- A simple (but suboptimal) solution: Each node should be close to its neighbors

$$\max_{\boldsymbol{e}_h, \boldsymbol{e}_t} \sum_{(h,r,t)\in\mathcal{G}} \frac{\exp(\boldsymbol{e}_h^T \boldsymbol{e}_t)}{\sum_{t'} \exp(\boldsymbol{e}_h^T \boldsymbol{e}_{t'})}$$

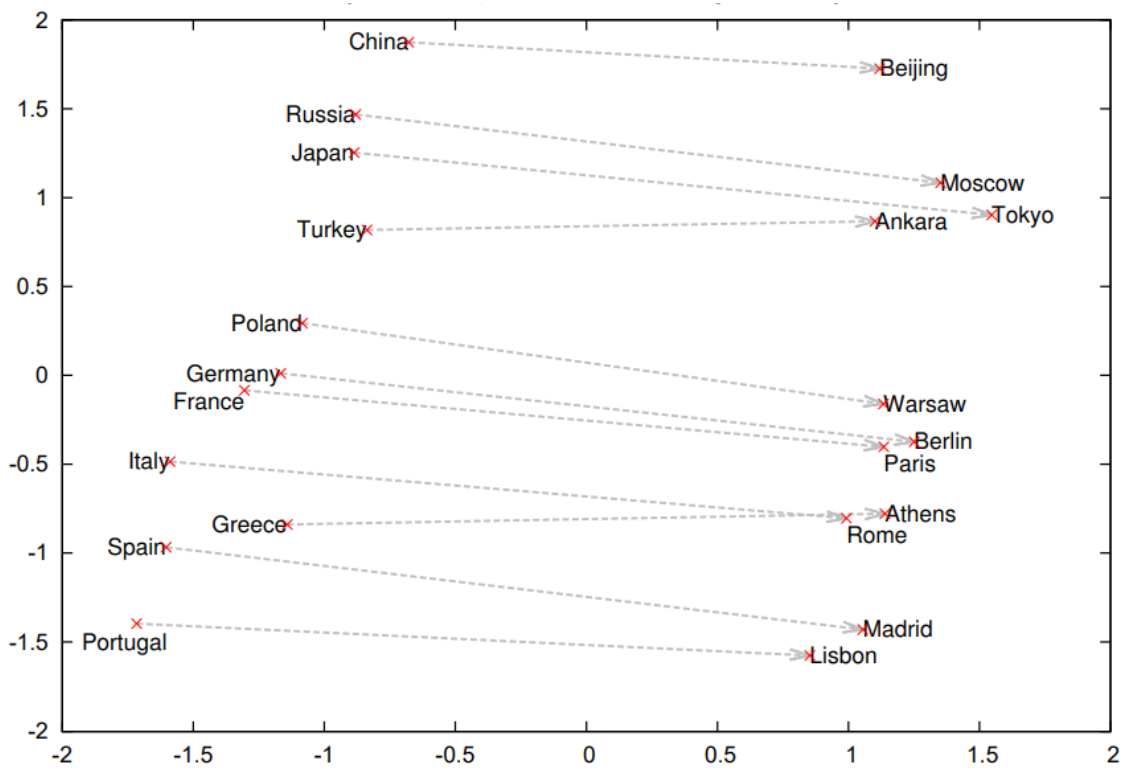
  - Limitation: It ignores different relations

# TransE [Bordes et al., NIPS 2013]

- Idea: Model relations as translations

| head entity $h$ | relation $r$ | tail entity $t$ |
|---|---|---|
| *Paris* | is capital of | *France* |

$$\boldsymbol{e}_h + \boldsymbol{e}_r \approx \boldsymbol{e}_t$$



# TransE [Bordes et al., NIPS 2013]

- Idea: Model relations as translations
- Training:

| head entity $h$ | relation $r$ | tail entity $t$ |
|---|---|---|
| *Paris* | is capital of | *France* |

Minimize $\|\boldsymbol{e}_h + \boldsymbol{e}_r - \boldsymbol{e}_t\|$

| head entity $h$ | relation $r$ | "corrupted" tail entity $t'$ |
|---|---|---|
| *Paris* | is capital of | *UK* |

Maximize $\|\boldsymbol{e}_h + \boldsymbol{e}_r - \boldsymbol{e}_{t'}\|$

(but stop when it becomes sufficiently larger than $\|\boldsymbol{e}_h + \boldsymbol{e}_r - \boldsymbol{e}_t\|$)

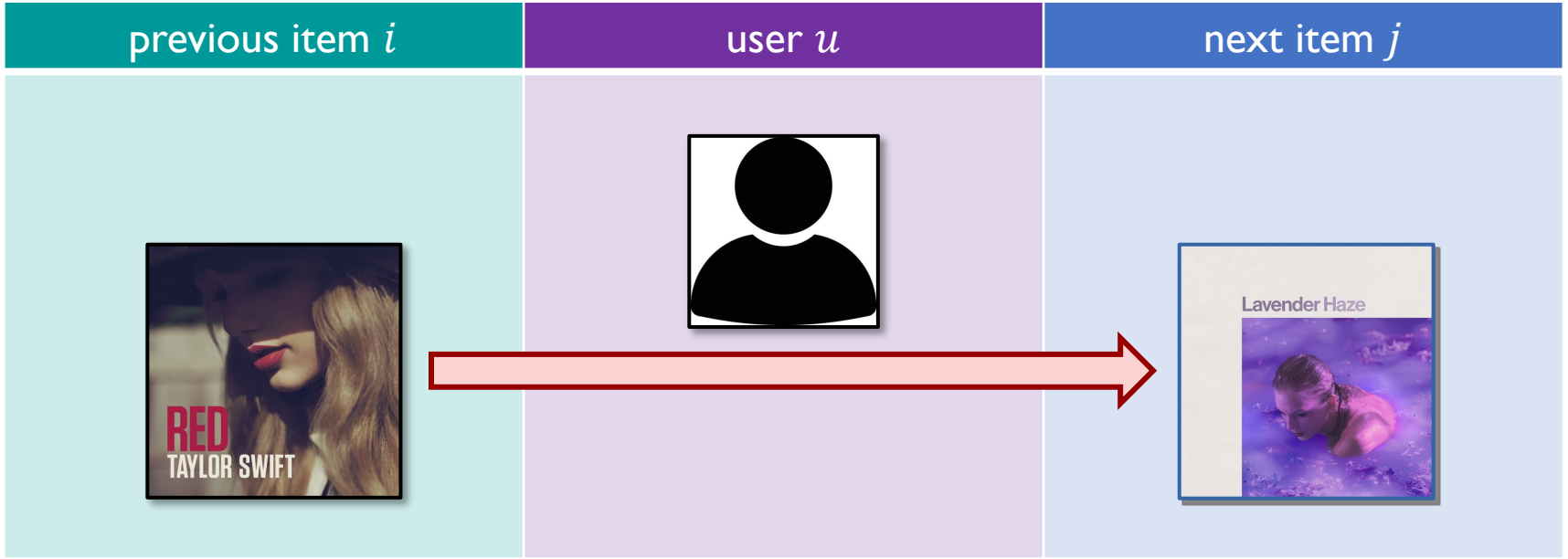
- Testing:

| head entity $h$ | relation $r$ | tail entity $t$ |
|---|---|---|
| *Ottawa* | is capital of | *?* |

Find the nearest neighbor of $\boldsymbol{e}_h + \boldsymbol{e}_r$

# Translation-Based Recommendation

- Items as points
- Users as translational vectors

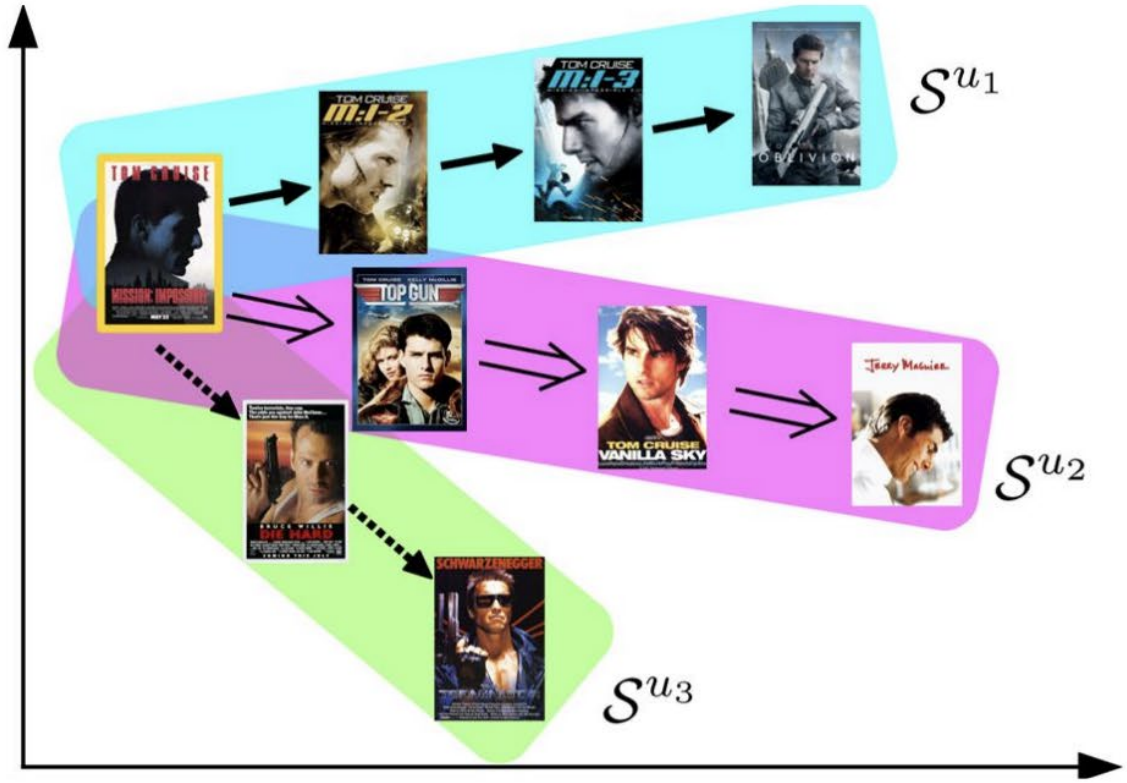



$$\boldsymbol{e}_i + \boldsymbol{e}_u \approx \boldsymbol{e}_j$$

# Translation-Based Recommendation

- Items as points
- Users as translational vectors
  - E.g., the songs progressively become more up-tempo, as the user is at a party
    - $\boldsymbol{e}_u$:"the next song is more up-tempo than the previous song"
  - E.g., the songs gradually become more relaxing, as the user is going to sleep
    - $\boldsymbol{e}_u$:"the next song is more relaxing than the previous song"

# Translation-Based Recommendation




User $u_1$

User $u_2$

User $u_3$

**Translation operation:**

$$\overrightarrow{prev.\ item} + \overrightarrow{user} \approx \overrightarrow{next\ item}$$

- Testing time: Find the nearest neighbor of $\boldsymbol{e}_i + \boldsymbol{e}_u$

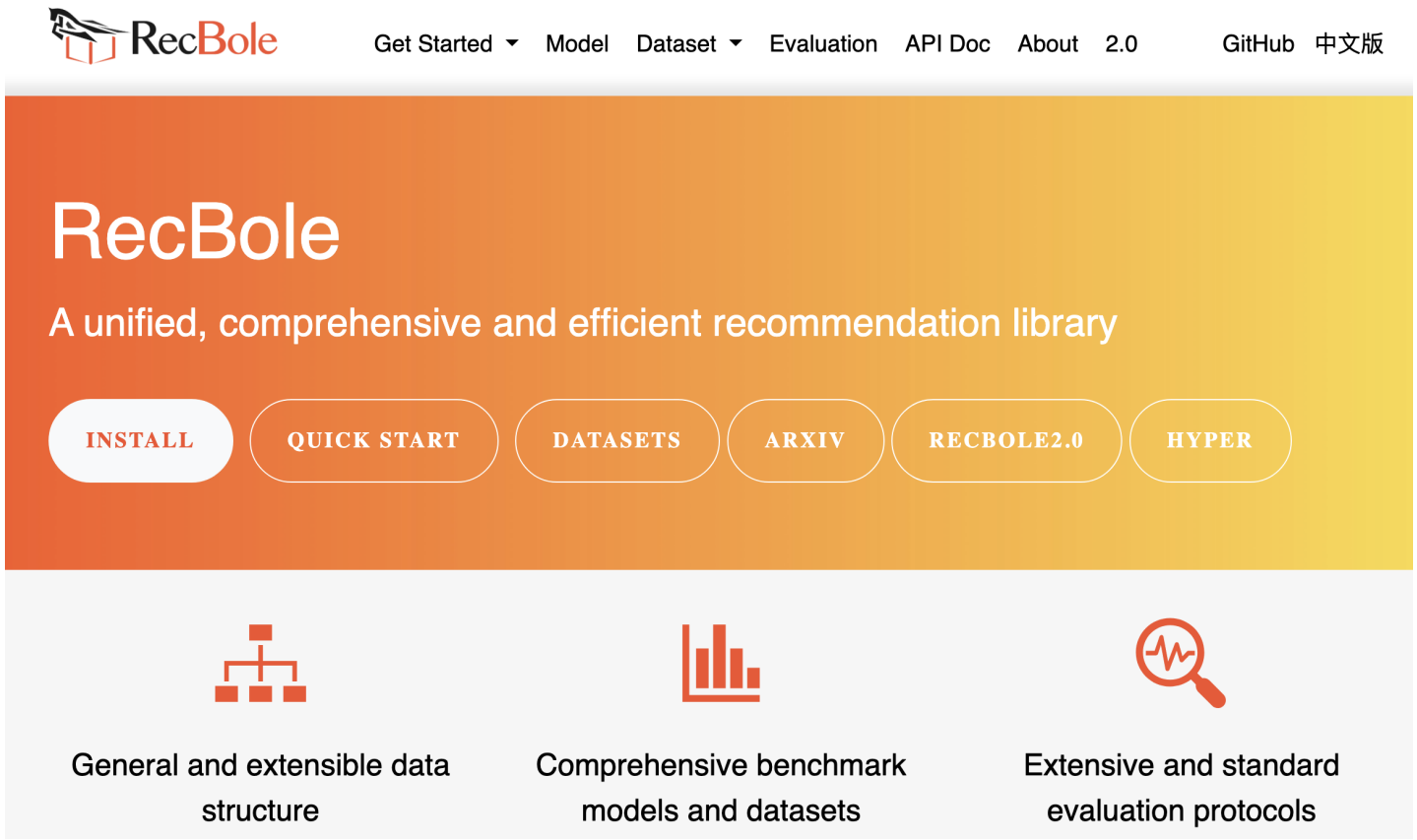
# Performance

- Outperforms BPR-MF
- No comparisons with Fossil

| Dataset | Metric | PopRec | BPR-MF | FMC | FPMC | $HRM_{avg}$ | $HRM_{max}$ | PRME | TransRec $\mathcal{L}_1$ | TransRec $\mathcal{L}_2$ | %Improv. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Epinions* | *AUC* | 0.4576 | 0.5523 | 0.5537 | 0.5517 | 0.6060 | 0.5617 | 0.6117 | 0.6063 | 0.6133 | 0.3% |
| | *Hit@50* | 3.42% | 3.70% | 3.84% | 2.93% | 3.44% | 2.79% | 2.51% | 3.18% | 4.63% | 20.6% |
| *Automotive* | *AUC* | 0.5870 | 0.6342 | 0.6438 | 0.6427 | 0.6704 | 0.6556 | 0.6469 | 0.6779 | 0.6868 | 2.5% |
| | *Hit@50* | 3.84% | 3.80% | 2.32% | 3.11% | 4.47% | 3.71% | 3.42% | 5.07% | 5.37% | 20.1% |
| *Google* | *AUC* | 0.5391 | 0.8188 | 0.7619 | 0.7740 | 0.8640 | 0.8102 | 0.8252 | 0.8359 | 0.8691 | 0.6% |
| | *Hit@50* | 0.32% | 4.27% | 3.54% | 3.99% | 3.55% | 4.59% | 5.07% | 6.37% | 6.84% | 32.5% |
| *Office* | *AUC* | 0.6427 | 0.6979 | 0.6867 | 0.6866 | 0.6981 | 0.7005 | 0.7020 | 0.7186 | 0.7302 | 4.0% |
| | *Hit@50* | 1.66% | 4.09% | 2.66% | 2.97% | 5.50% | 4.17% | 6.20% | 6.86% | 6.51% | 10.7% |
| *Toys* | *AUC* | 0.6240 | 0.7232 | 0.6645 | 0.7194 | 0.7579 | 0.7258 | 0.7261 | 0.7442 | 0.7590 | 0.2% |
| | *Hit@50* | 1.69% | 3.60% | 1.55% | 4.41% | 5.25% | 3.74% | 4.80% | 5.46% | 5.44% | 4.0% |
| *Clothing* | *AUC* | 0.6189 | 0.6508 | 0.6640 | 0.6646 | 0.7057 | 0.6862 | 0.6886 | 0.7047 | 0.7243 | 2.6% |
| | *Hit@50* | 1.11% | 1.05% | 0.57% | 0.51% | 1.70% | 1.15% | 1.00% | 1.76% | 2.12% | 24.7% |
| *Cellphone* | *AUC* | 0.6959 | 0.7569 | 0.7347 | 0.7375 | 0.7892 | 0.7654 | 0.7860 | 0.7988 | 0.8104 | 2.7% |
| | *Hit@50* | 4.43% | 5.15% | 3.23% | 2.81% | 8.77% | 6.32% | 6.95% | 9.46% | 9.54% | 8.8% |
| *Games* | *AUC* | 0.7495 | 0.8517 | 0.8407 | 0.8523 | 0.8776 | 0.8566 | 0.8597 | 0.8711 | 0.8815 | 0.4% |
| | *Hit@50* | 5.17% | 10.93% | 13.93% | 12.29% | 14.44% | 12.86% | 14.22% | 16.61% | 16.44% | 15.0% |
| *Electronics* | *AUC* | 0.7837 | 0.8096 | 0.8158 | 0.8082 | 0.8212 | 0.8148 | 0.8337 | 0.8457 | 0.8484 | 1.8% |
| | *Hit@50* | 4.62% | 2.98% | 4.15% | 2.82% | 4.09% | 2.59% | 3.07% | 4.89% | 5.19% | 12.3% |
| *Foursquare* | *AUC* | 0.9168 | 0.9511 | 0.9463 | 0.9479 | 0.9559 | 0.9523 | 0.9565 | 0.9631 | 0.9651 | 0.9% |
| | *Hit@50* | 55.60% | 60.03% | 63.00% | 64.53% | 60.75% | 61.60% | 65.32% | 66.12% | 67.09% | 2.7% |
| *Flixter* | *AUC* | 0.9459 | 0.9722 | 0.9568 | 0.9718 | 0.9695 | 0.9687 | 0.9728 | 0.9727 | 0.9750 | 0.2% |
| | *Hit@50* | 11.92% | 21.58% | 22.23% | 33.11% | 32.34% | 30.88% | 40.81% | 35.52% | 35.02% | -13.0% |

# Homework 4 (Part 1): Compare Fossil and TransRec

- No need to implement them yourself!
- The RecBole library: https://recbole.io/



# Large Language Models Basics

# We are finally here!

- ✅ Phase 1: Search Engines
  - basics, Boolean and ranked retrieval, link analysis, evaluation, learning to rank (ML + ranking), …
- ✅ Phase 2: Recommender Systems
  - basics, non-personalized recommendation, collaborative filtering, matrix factorization, implicit recommendation, …
- ✅ Phase 3: From Foundations to Modern Methods
  - embedding learning, Transformer, "small" language models, … (for search and recommendation)
- Phase 4: Large Language Models (!!)

# NLP before the Era of Large Language Models

- Given an NLP task
  - Step 1: Find/annotate task-specific training data (e.g., 10,000 training samples)
  - Step 2: Design a task-specific model (e.g., 3-layer recurrent neural networks with …)
  - Step 3: Use the annotated data to train the model

*Math Word Problem*: (*John buys 20 cards and 1/4 are uncommon. How many uncommon cards did he get?*, *5*) → Math Model 

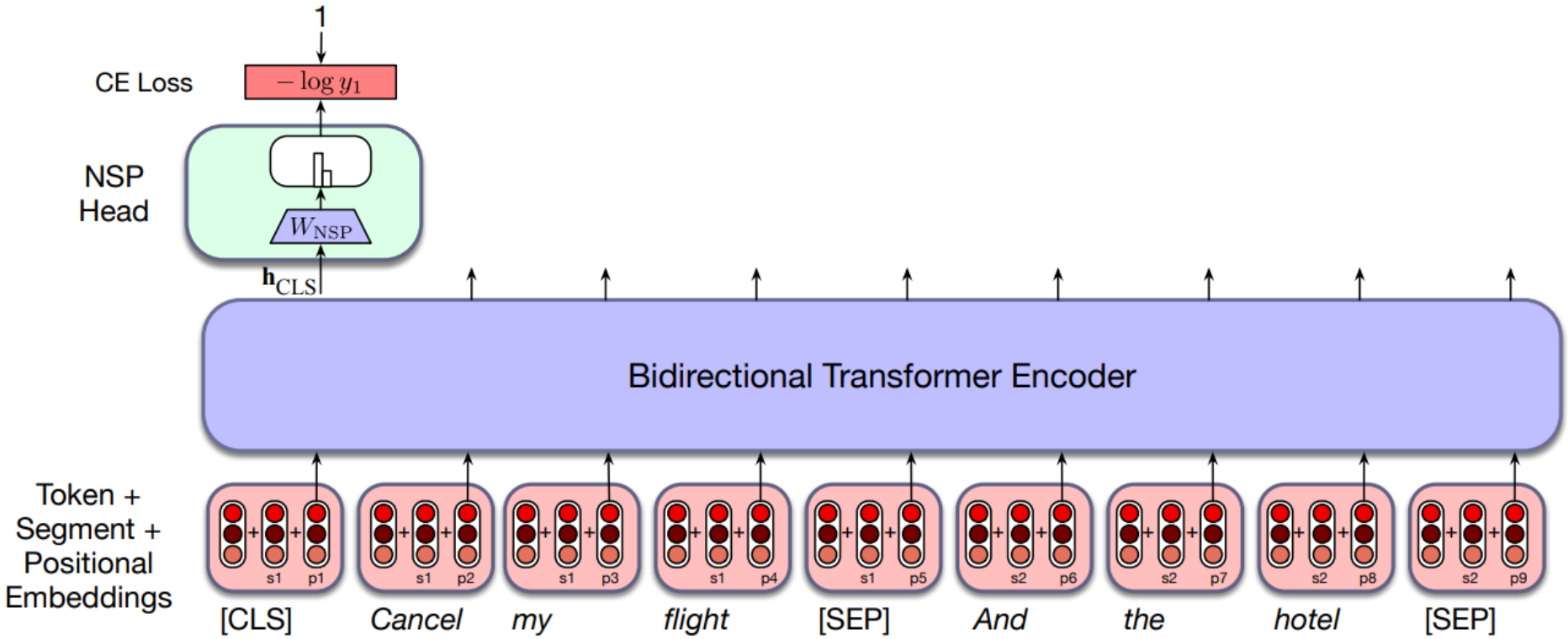
*Paper Classification*: (*Rapid chromatographic technique for preparative separations with moderate resolution*, *Organic Chemistry*) → Classification Model 

*Information Extraction*: (*in rats, nitrofurantoin causes pulmonary toxicity*, *[pulmonary toxicity, DISEASE]*) → IE Model 

*Question Answering*: (*Who formulated the zeroth law of thermodynamics?*, *Ralph H. Fowler*) → QA Model 

# Recap: BERT [Devlin et al., NAACL 2019]

- BERT has already learned knowledge, linguistic patterns, and other generally useful information for various NLP tasks from the entire Wikipedia and BookCorpus.
- However, it cannot be directly used for many downstream tasks without any further fine-tuning.



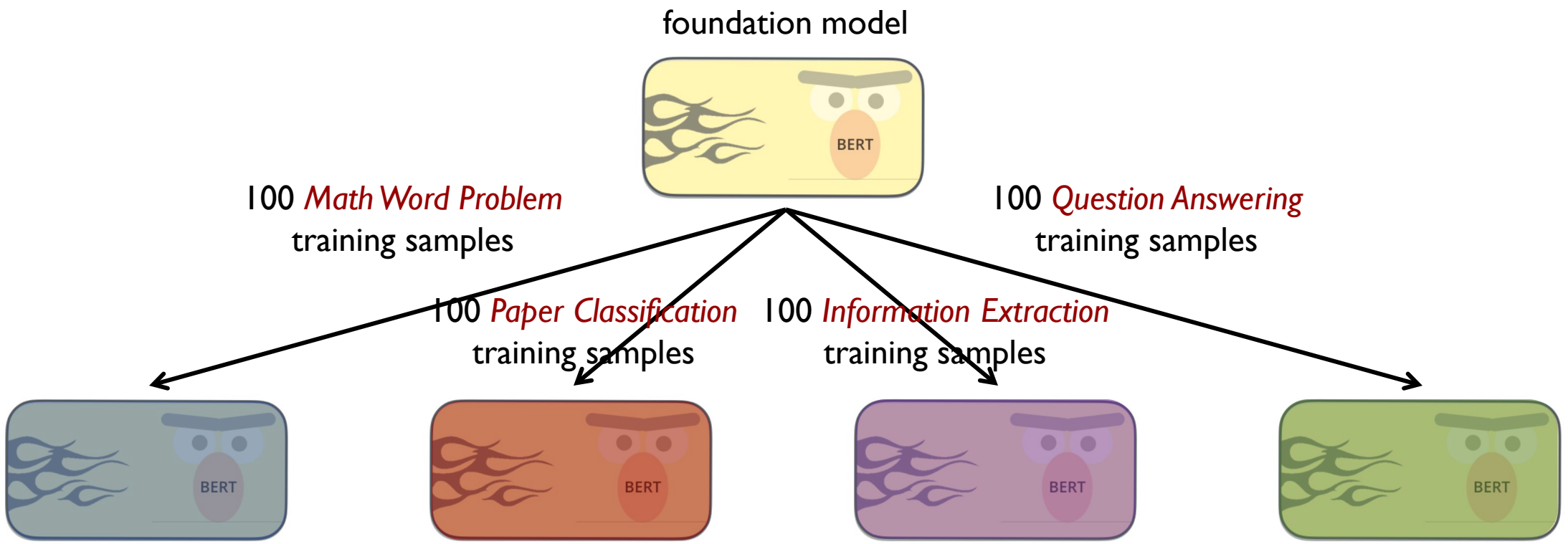
# How to fine-tune BERT for different NLP tasks?

- *Math Word Problem*: Input the entire math word problem into BERT, get the [CLS] token embedding, and perform regression by training a regression layer
- *Paper Classification*: Input the paper into BERT, get the [CLS] token embedding, and perform topic classification by training a classification layer
- *Information Extraction*: Input the text into BERT, get the embedding of each token, and classify each token

| *in* | *rats* | *,* | *nitrofurantoin* | *causes* | *pulmonary* | *toxicity* |
|---|---|---|---|---|---|---|
| *None* | *None* | *None* | *B-Chemical* | *None* | *B-Disease* | *I-Disease* |

- Suppose you train a math word problem solver from scratch (with all parameters initialized randomly), it might require around 10,000 training samples to achieve good performance.
- In contrast, fine-tuning BERT typically needs far fewer training samples (for example, around 100) to reach comparable performance.

# The BERT Revolution



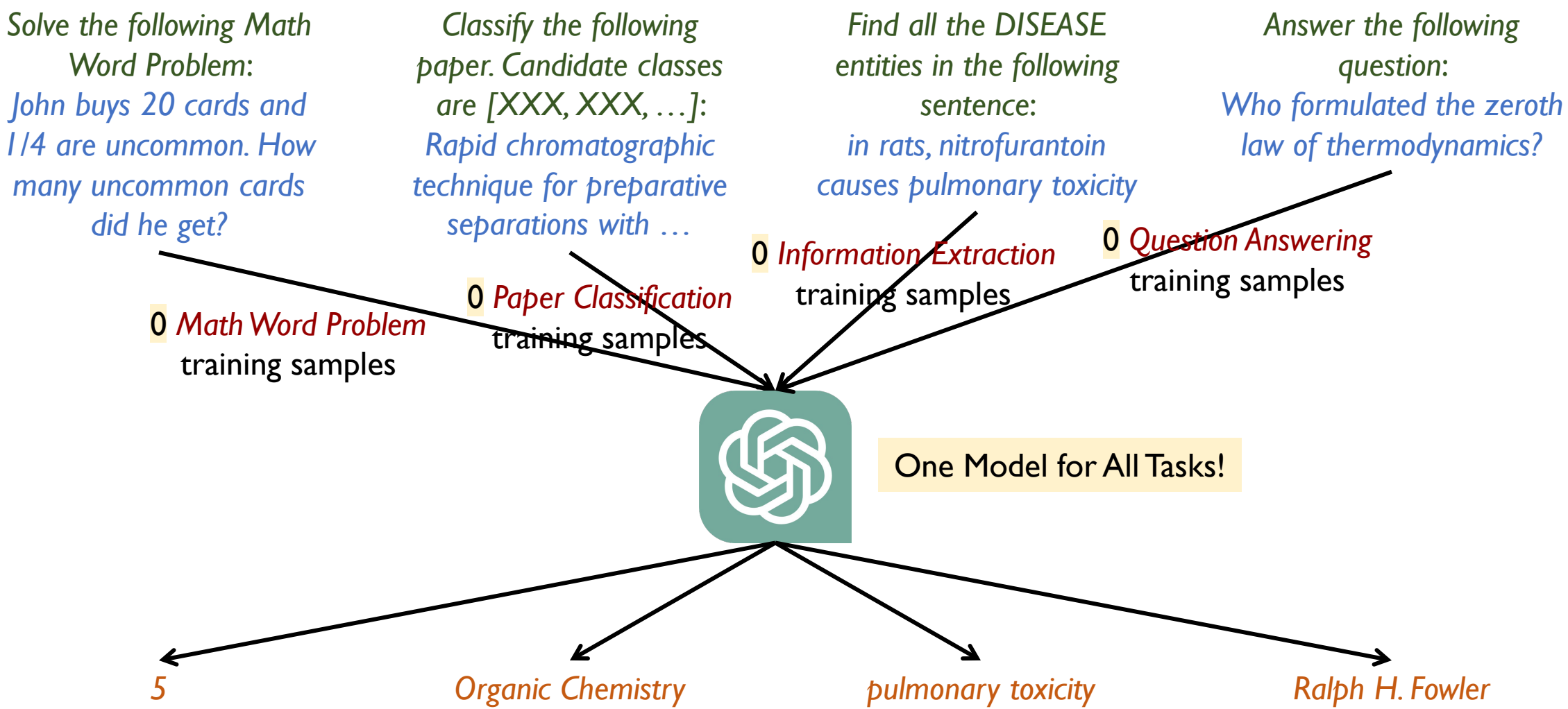

*Math Word Problem*: (*John buys 20 cards and 1/4 are uncommon. How many uncommon cards did he get?*, *5*)

*Paper Classification*: (*Rapid chromatographic technique for preparative separations with moderate resolution*, *Organic Chemistry*)

*Information Extraction*: (*in rats, nitrofurantoin causes pulmonary toxicity*, *[pulmonary toxicity, DISEASE]*)

*Question Answering*: (*Who formulated the zeroth law of thermodynamics?*, *Ralph H. Fowler*)

# But today's ChatGPT is far more powerful!



# Let's take a step back first

The task instruction is optional if you have 5 training samples.

5 *Math Word Problem* training samples

*John buys 20 cards and 1/4 are uncommon. How many uncommon cards did he get?*

5 *Paper Classification* training samples

*Rapid chromatographic technique for preparative separations with moderate resolution*

5 *Information Extraction* training samples

*in rats, nitrofurantoin causes pulmonary toxicity*

5 *Question Answering* training samples

*Who formulated the zeroth law of thermodynamics?*



One Model for All Tasks!

*5*

*Organic Chemistry*

*pulmonary toxicity*

*Ralph H. Fowler*

# GPT-3 [Brown et al., NeurIPS 2020]

## Language Models are Few-Shot Learners

**Tom B. Brown*** **Benjamin Mann*** **Nick Ryder*** **Melanie Subbiah***

**lakantan** **Pranav Shyam** **Girish Sastry**

**t-Voss** **Gretchen Krueger** **Tom Henighan**

**iegler** **Jeffrey Wu** **Clemens Winter**

**gler** **Mateusz Litwin** **Scott Gray**

**Christopher Berner**

**Ilya Sutskever** **Dario Amodei**

I

Language models are few-shot learners

T Brown, B Mann, N Ryder, M Subbiah, JD Kaplan, P Dhariwal, A Neelakantan, P Shyam...

Advances in neural information processing systems, 2020 • proceedings.neurips.cc

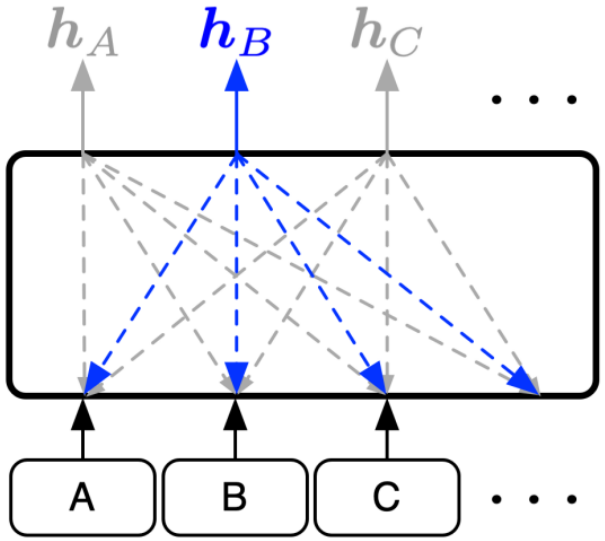
**Abstract**

We demonstrate that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even becoming competitive with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely

SHOW MORE

☆ Save 99 Cite Cited by 85129 Related articles All 37 versions »

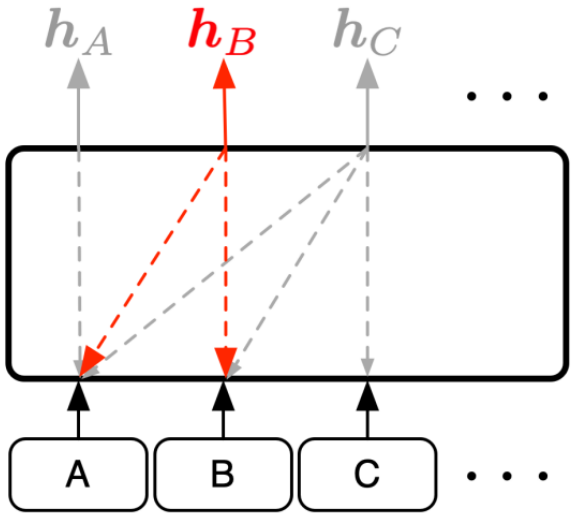
# Recap: Two Types of Transformer Architecture




*every token attends to all tokens*

**Bidirectional Self-Attention**

| | | | |
|---|---|---|---|
| q1·k1 | q1·k2 | q1·k3 | q1·k4 |
| q2·k1 | q2·k2 | q2·k3 | q2·k4 |
| q3·k1 | q3·k2 | q3·k3 | q3·k4 |
| q4·k1 | q4·k2 | q4·k3 | q4·k4 |




*every token attends to its previous tokens*

**Unidirectional Self-Attention**

| | | | |
|---|---|---|---|
| q1·k1 | −∞ | −∞ | −∞ |
| q2·k1 | q2·k2 | −∞ | −∞ |
| q3·k1 | q3·k2 | q3·k3 | −∞ |
| q4·k1 | q4·k2 | q4·k3 | q4·k4 |

# What if we pre-train a Transformer decoder?

- With only one task – next token prediction – on web-scale corpora

$$\sum_i p(\, w_i \mid w_1, w_2, \ldots, w_{i-1} \,;\, \Theta \,)$$

next token | previous tokens | model parameters

- Most NLP tasks can "reduce" to next token prediction.
  - *Math*: {*Input to the model*} John buys 20 cards and 1/4 are uncommon. How many uncommon cards did he get? The answer is
    - {*Model predicts the next token(s)*} 11
  - *Classification*: {*Input to the model*} (paper title) Rapid chromatographic technique for preparative separations with moderate resolution => (label)
    - {*Model predicts the next token(s)*} Organic Chemistry

# What if we pre-train a Transformer decoder?

- With only one task – next token prediction – on web-scale corpora

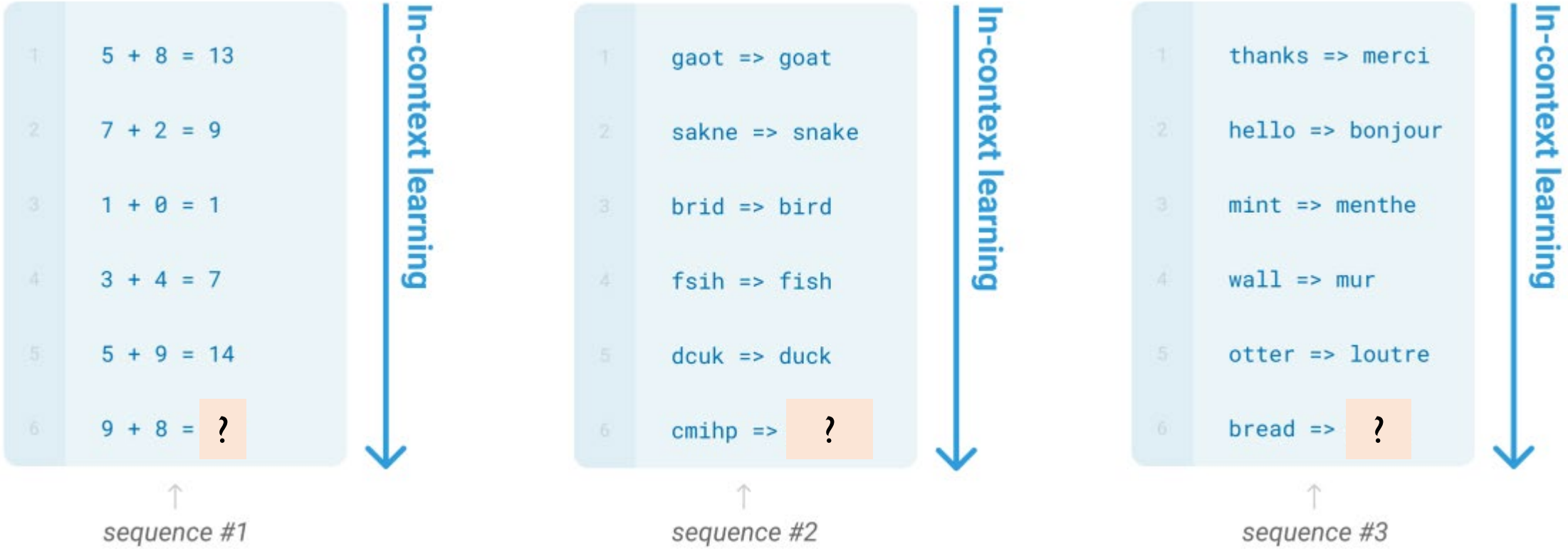
$$\sum_i p(\, w_i \mid w_1, w_2, \ldots, w_{i-1} \,;\, \Theta \,)$$

next token | previous tokens | model parameters

- Most NLP tasks can "reduce" to next token prediction.
  - *Information Extraction*: {*Input to the model*} (text) in rats, nitrofurantoin causes pulmonary toxicity. => (entity, type)
    - {*Model predicts the next token(s)*} pulmonary toxicity, disease
  - *Question Answering*: {*Input to the model*} Who formulated the zeroth law of thermodynamics?
    - {*Model predicts the next token(s)*} Ralph H. Fowler

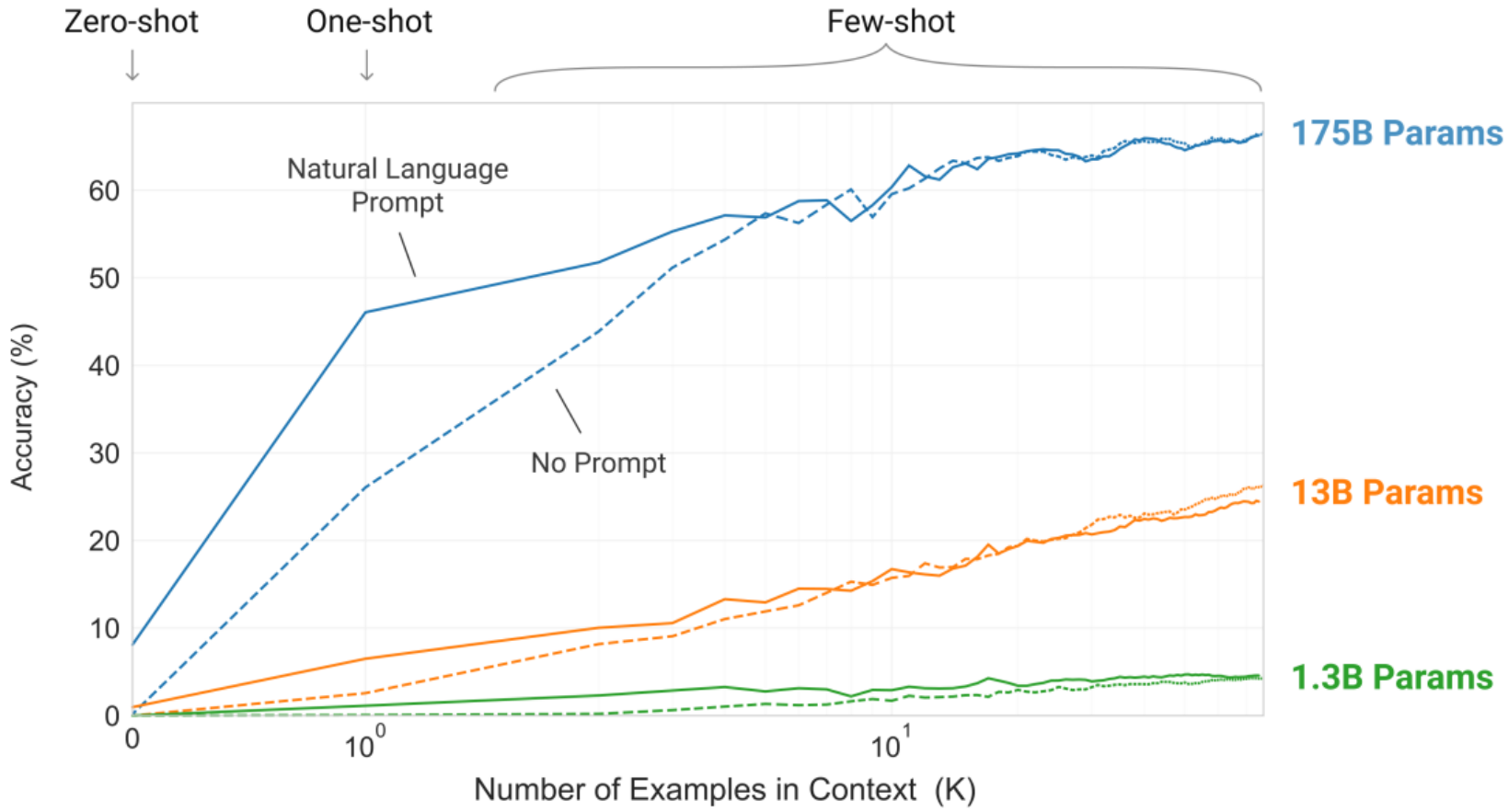
# GPT-3 can perform each task with just a few examples

- The model may acquire a broad set of skills and pattern recognition abilities during pre-training. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. – "In-context learning"



# Can a model be that "smart"?

- Only if it is big enough!




GPT-3

BERT-Base: 0.11B
BERT-Large: 0.34B

Can we make our demo examples more insightful?
(Prompt Engineering)

# Chain-of-Thought Prompting [Wei et al., NeurIPS 2022]

**Chain-of-Thought Prompting Elicits Reasoning in Large Language Models**

Chain-of-thought prompting elicits reasoning in large language models

J Wei, X Wang, D Schuurmans, M Bosma, F Xia, E Chi, QV Le, D Zhou

Advances in neural information processing systems, 2022 • proceedings.neurips.cc

**Abstract**

We explore how generating a chain of thought---a series of intermediate reasoning steps---significantly improves the ability of large language models to perform complex reasoning. In particular, we show how such reasoning abilities emerge naturally in sufficiently large language models via a simple method called chain of thought prompting, where a few chain of thought demonstrations are provided as exemplars in prompting. Experiments on three large language models show that chain of thought prompting improves performance

SHOW MORE

☆ Save 99 Cite Cited by 40902 Related articles All 28 versions »

**le Schuurmans** **Maarten Bosma**

**Chi** **Quoc V. Le** **Denny Zhou**

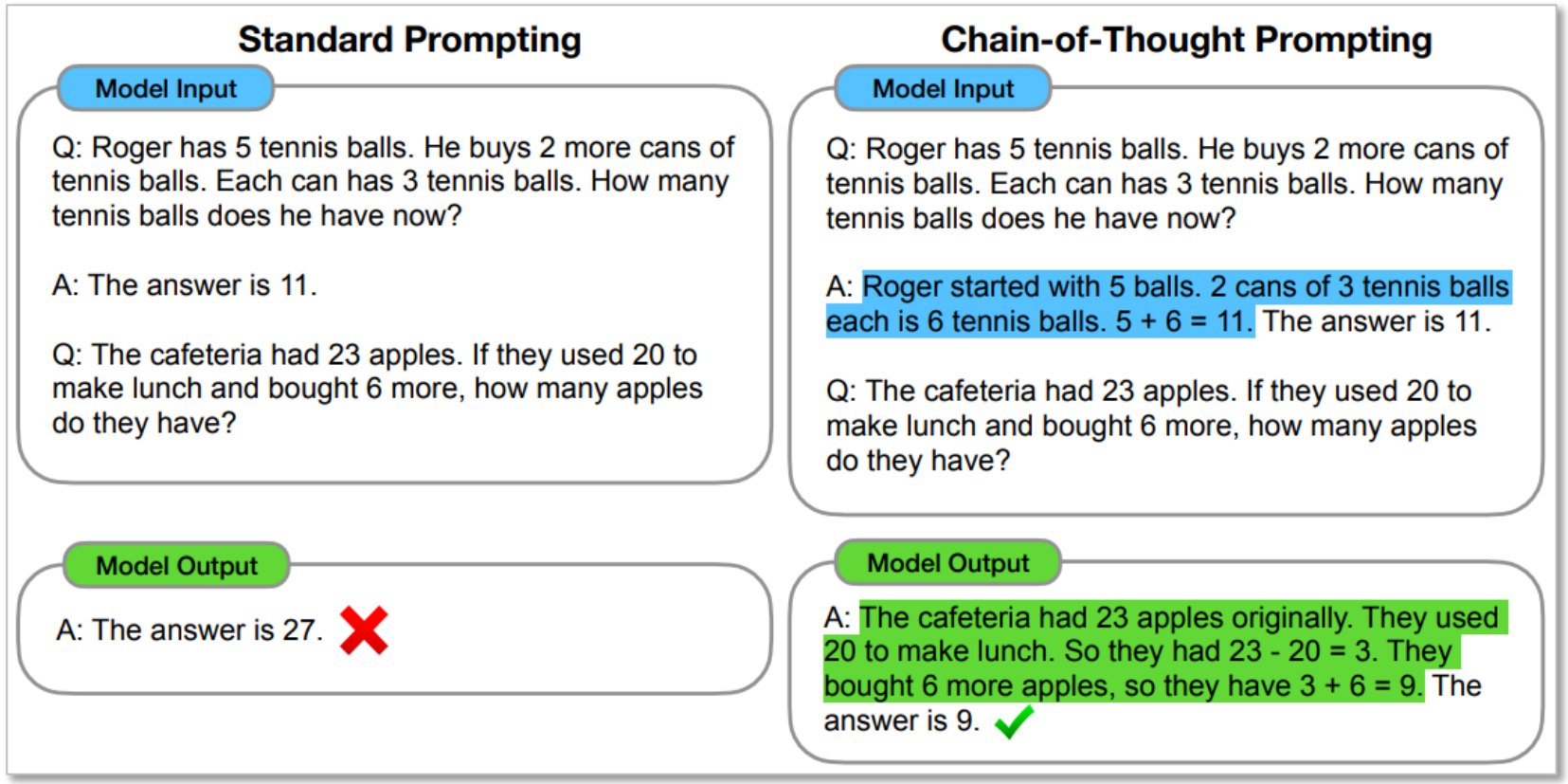
ch, Brain Team
hou}@google.com

**ract**

*ught*—a series of intermediate reasoning
y of large language models to perform
ow how such reasoning abilities emerge
dels via a simple method called *chain-of-*
thought demonstrations are provided as

# Chain-of-Thought Prompting

- Add a series of intermediate reasoning steps in the demonstration examples(s)
- Get the model to explain its reasoning steps before making an answer



# Self-Consistency [Wang et al., ICLR 2023]

## SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

**Xuezhi Wang**[†‡] **Jason Wei**[†] **Dale Schuurmans**[†] **Quoc Le**[†] **Ed H. Chi**[†]
**Sharan Narang**[†] **Aakanksha Chowdhery**[†] **Denny Zhou**[†§]
[†]Google Research, Brain Team
[‡]xuezhiw@google.com, [§]dennyzhou@google.com

Self-consistency improves chain of thought reasoning in language models
X Wang, J Wei, D Schuurmans, Q Le, E Chi, S Narang, A Chowdhery, D Zhou
arXiv preprint arXiv:2203.11171, 2022 • arxiv.org

Chain-of-thought prompting combined with pre-trained large language models has achieved encouraging results on complex reasoning tasks. In this paper, we propose a new decoding strategy, self-consistency, to replace the naive greedy decoding used in chain-of-thought prompting. It first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths. Self-consistency leverages the intuition that a complex

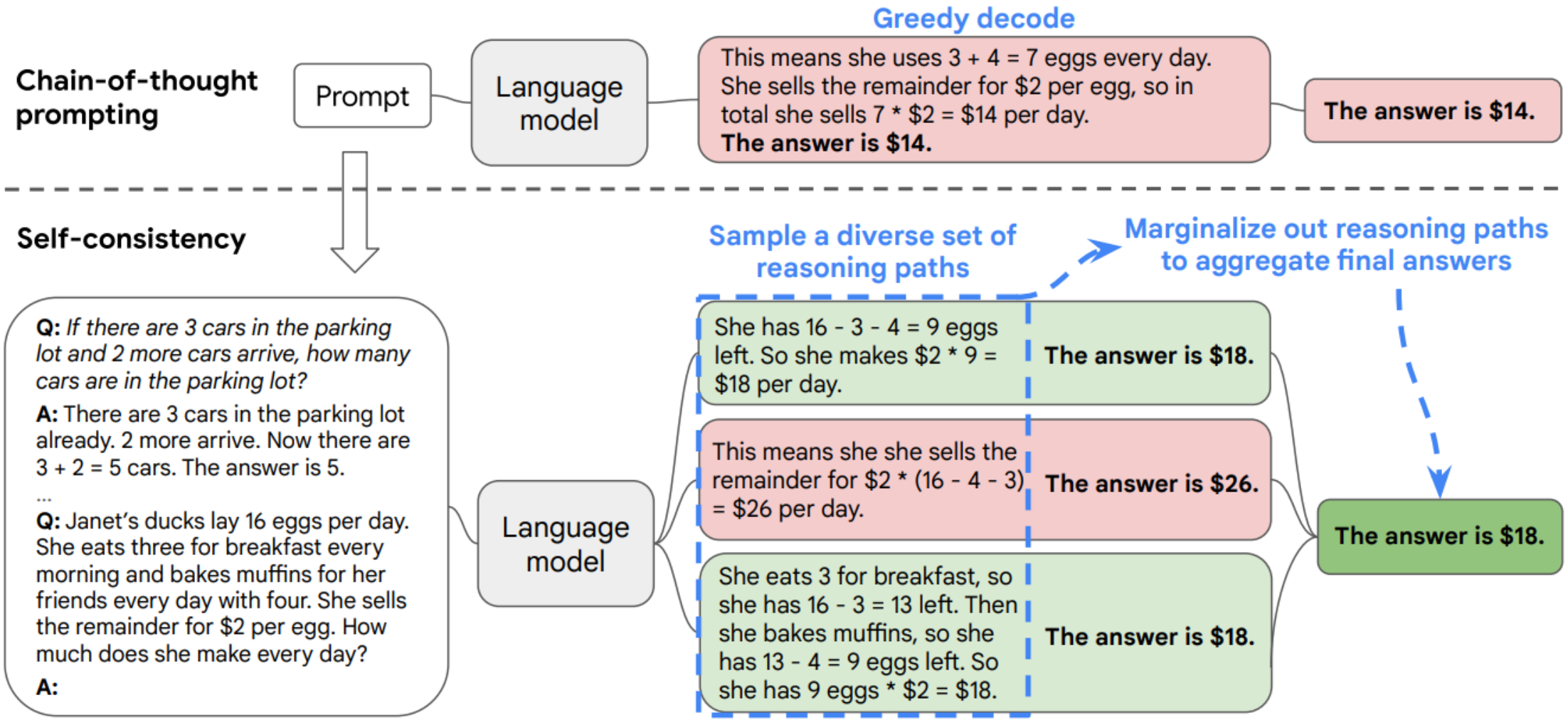
SHOW MORE

☆ Save 99 Cite Cited by 6239 Related articles All 6 versions »

ABSTRACT

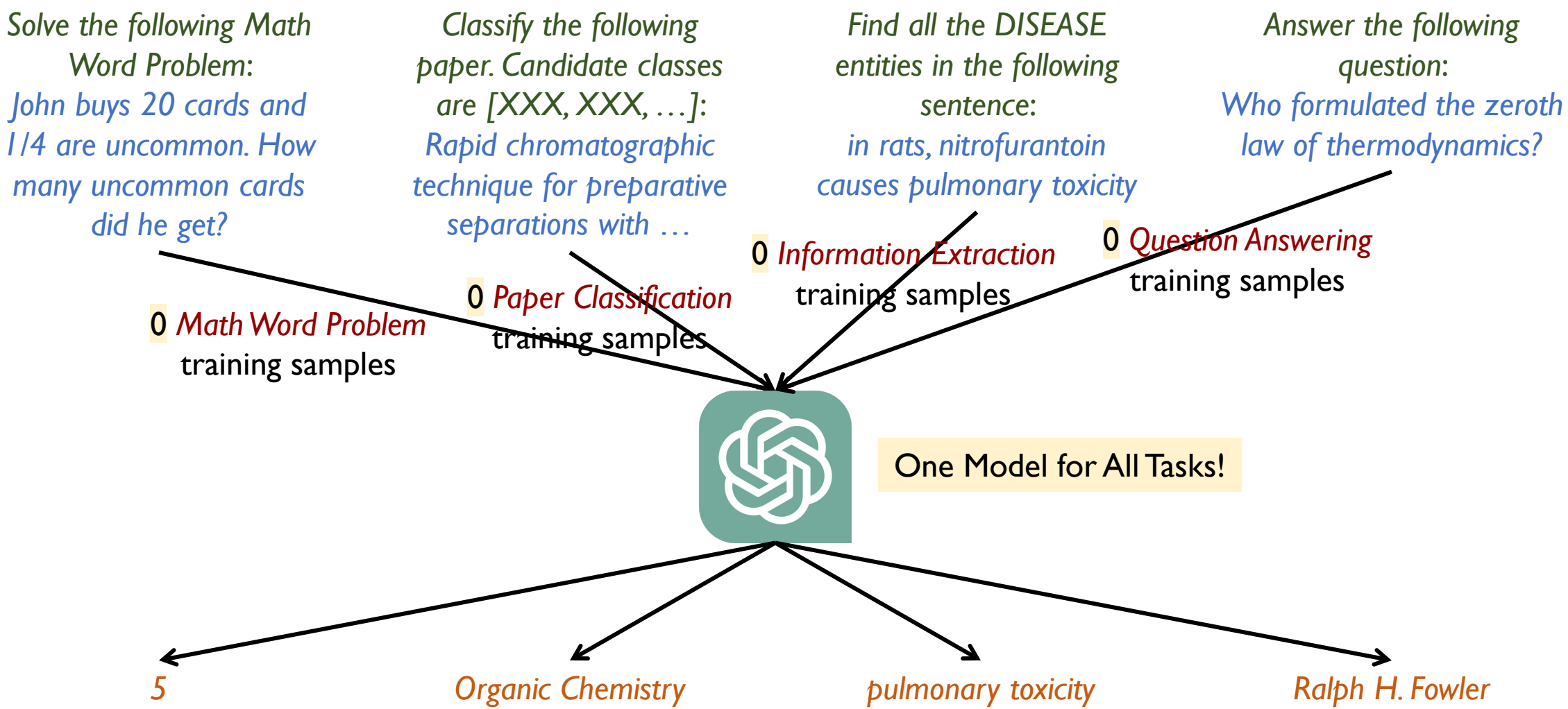
ed with pre-trained large language models has
plex reasoning tasks. In this paper, we propose
*istency*, to replace the naive greedy decoding
It first samples a diverse set of reasoning paths
e, and then selects the most consistent answer
easoning paths. Self-consistency leverages the
oblem typically admits multiple different ways
ect answer. Our extensive empirical evaluation
he performance of chain-of-thought prompting
opular arithmetic and commonsense reasoning
17.9%), SVAMP (+11.0%), AQuA (+12.2%),
lenge (+3.9%).

# Self-Consistency



# From Few-Shot to Zero-Shot (Instruction Tuning and Alignment)

# Our Final Goal



# From Few-Shot to Zero-Shot

**Few-shot**

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

**Zero-shot**

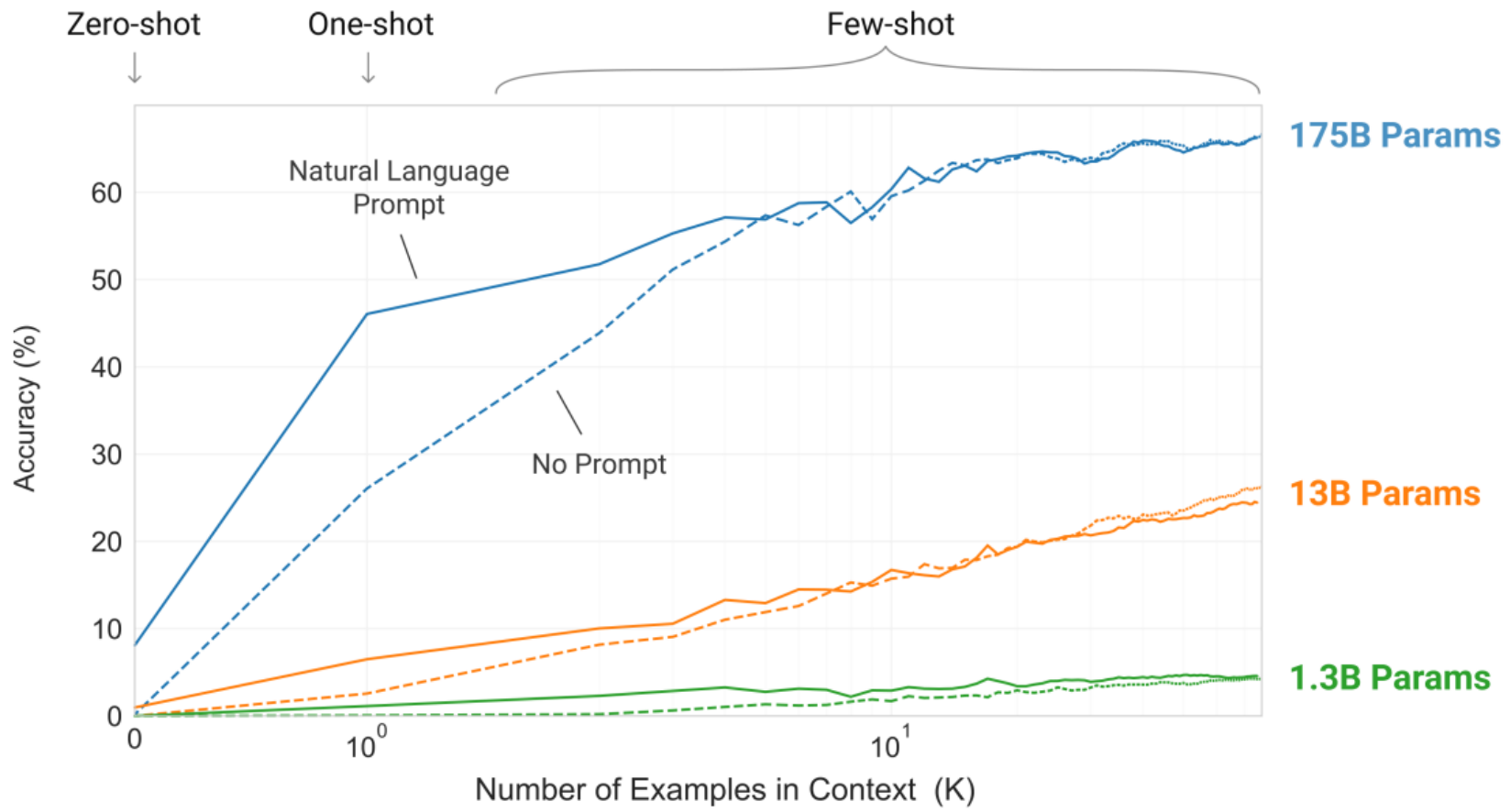
The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

# Is GPT-3 a zero-shot learner?

Task Instruction Only

Task Instruction + A Few Examples



# Why is the zero-shot setting hard?

- GPT-3 is not good at following an instruction to perform a new task.
  - Because it is never asked to do so during pre-training.
- How to solve this problem?
  - Tune the model to follow task instructions!

$$\sum_i p(\text{ ground truth output } | \text{ instruction, task input })$$

- How can we ensure that the model can handle unseen task instructions when it is actually used?
  - Let it see a sufficiently diverse set of task instructions during training so that it can generalize well
  - What really matters is the number of different tasks, not how many samples each task has
    - Find all available NLP benchmark datasets and convert them into the (instruction, input, output) format
    - Think of lots of creative new tasks and get people to label them

# FLAN [Wei et al., ICLR 2022]

## FINETUNED LANGUAGE MODELS ARE ZERO-SHOT LEARNERS

**Jason Wei*, Maarten Bosma*, Vincent Y. Zhao*, Kelvin Guu*, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le**

Google Research

Finetuned language models are zero-shot learners

J Wei, M Bosma, VY Zhao, K Guu, AW Yu, B Lester, N Du, AM Dai, QV Le

arXiv preprint arXiv:2109.01652, 2021 • arxiv.org

This paper explores a simple method for improving the zero-shot learning abilities of language models. We show that instruction tuning -- finetuning language models on a collection of tasks described via instructions -- substantially improves zero-shot performance on unseen tasks. We take a 137B parameter pretrained language model and instruction-tune it on over 60 NLP tasks verbalized via natural language instruction templates. We evaluate this instruction-tuned model, which we call FLAN, on unseen task

SHOW MORE

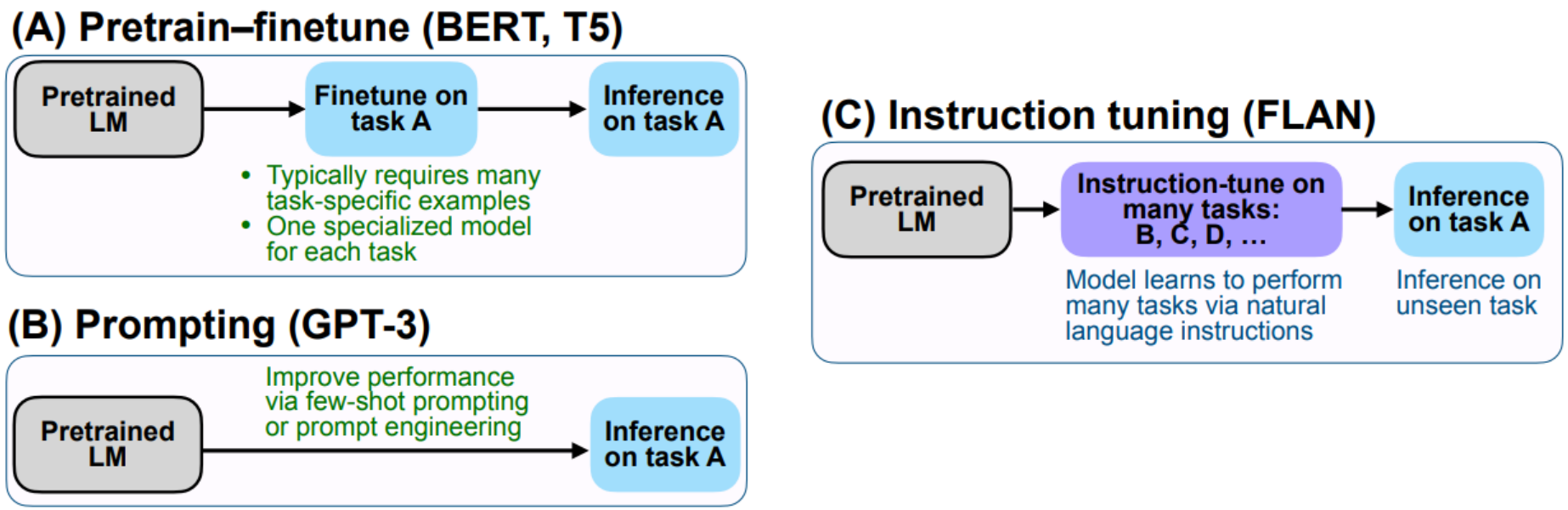
☆ Save 99 Cite Cited by 6779 Related articles All 11 versions »

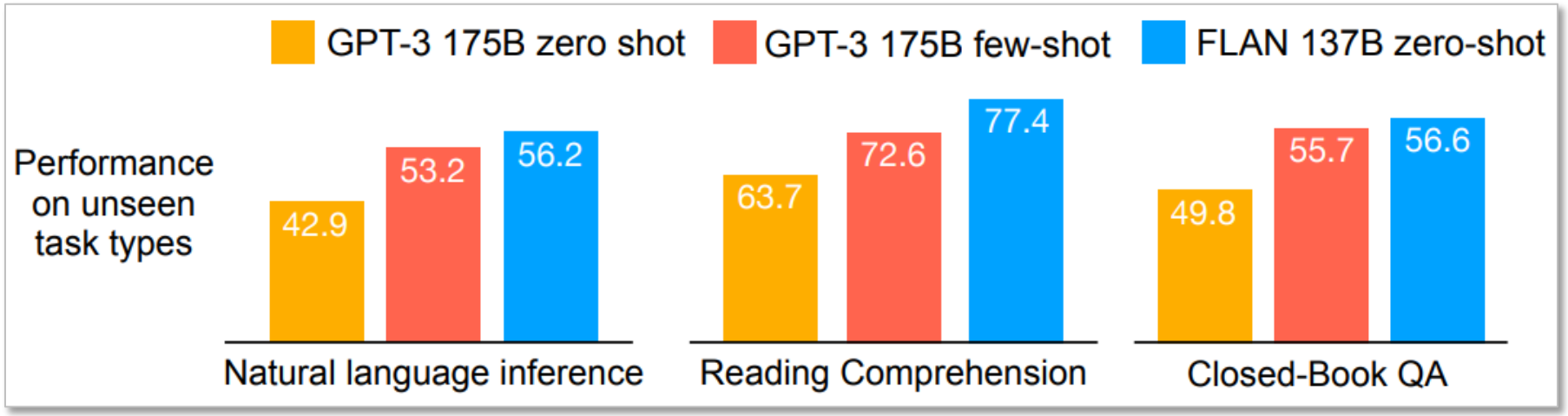
TRACT

r improving the zero-shot learning abilities
*uction tuning*—finetuning language models
instructions—substantially improves zero-

anguage model and instruction tune it on
atural language instruction templates. We
which we call FLAN, on unseen task types.
rmance of its unmodified counterpart and
f 25 datasets that we evaluate. FLAN even

# FLAN

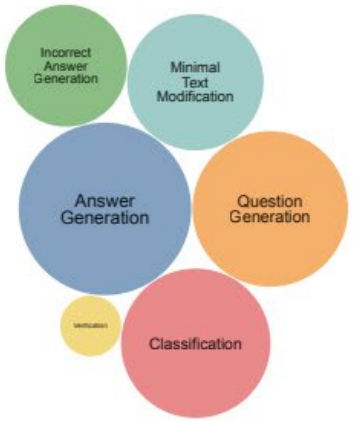

(A) Pretrain–finetune (BERT, T5)
Pretrained LM
Finetune on task A
Inference on task A
Typically requires many task-specific examples
One specialized model for each task
(B) Prompting (GPT-3)
Pretrained LM
Improve performance via few-shot prompting or prompt engineering
Inference on task A
(C) Instruction tuning (FLAN)
Pretrained LM
Instruction-tune on many tasks: B, C, D, …
Inference on task A
Model learns to perform many tasks via natural language instructions
Inference on unseen task


GPT-3 175B zero shot
GPT-3 175B few-shot
FLAN 137B zero-shot
Performance on unseen task types
42.9
53.2
56.2
63.7
72.6
77.4
49.8
55.7
56.6
Natural language inference
Reading Comprehension
Closed-Book QA

# Instruction Tuning: A Competition of Data Collection/Annotation



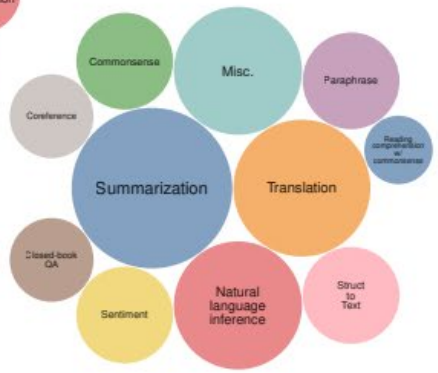

(a) SUP-NATINST (this work)



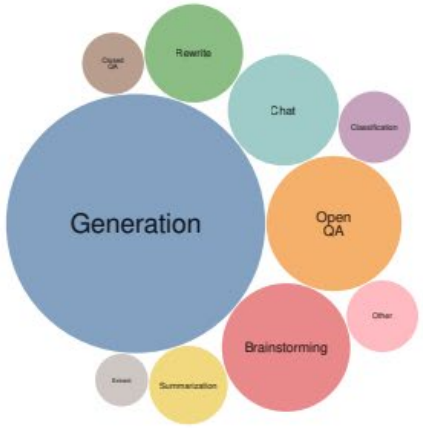

(b) NATINST




(c) PROMPTSOURCE (T0 subset)




(d) FLAN




(e) INSTRUCTGPT

Sup-NatInst: 1,616 tasks

FLAN: 62 tasks

# InstructGPT [Ouyang et al., NeurIPS 2022]

## Training language models to follow instructions with human feedback

Training language models to follow instructions with human feedback

L Ouyang, J Wu, X Jiang, D Almeida, C Wainwright, P Mishkin, C Zhang, S Agarwal...

Advances in neural information processing systems, 2022 • proceedings.neurips.cc

**Abstract**

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not aligned with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through a language model API, we collect a

SHOW MORE

☆ Save 99 Cite Cited by 33321 Related articles All 29 versions »

ogo Almeida* Carroll L. Wainwright*

Agarwal Katarina Slama Alex Ray

elton Luke Miller Maddie Simens

der Paul Christiano*†

Ryan Lowe*

I

# InstructGPT

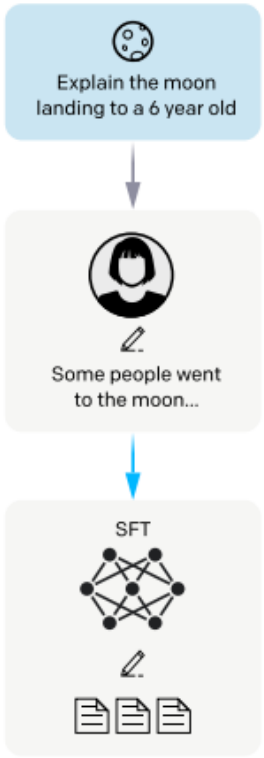
Step 1

**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.



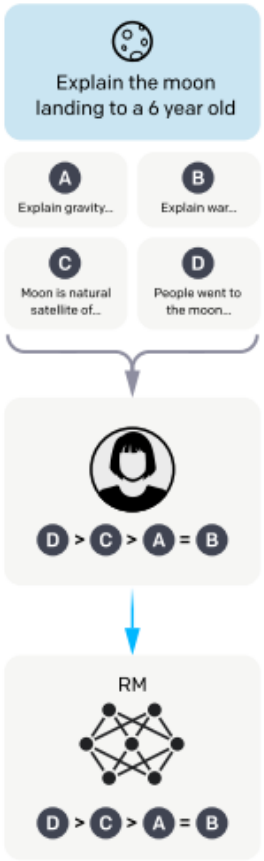

Step 2

**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

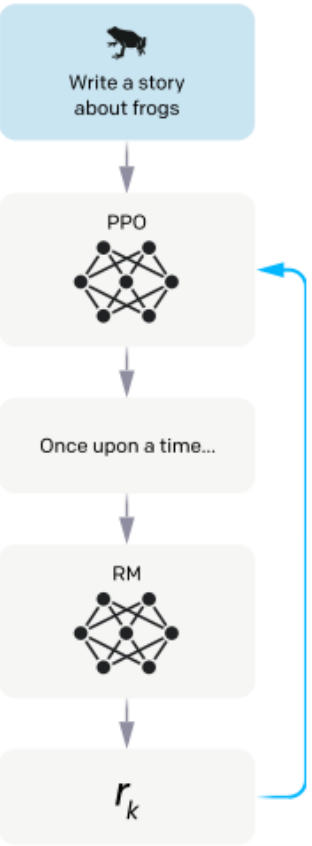
**Optimize a policy against the reward model using reinforcement learning.**

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.




Instruction Tuning

# InstructGPT

Train another model to imitate human preferences

Train GPT to satisfy the "human" model

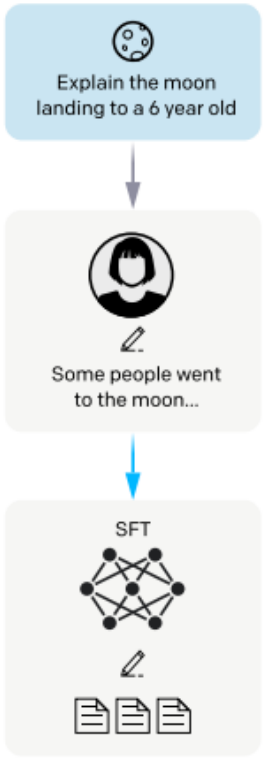
Step 1

**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.



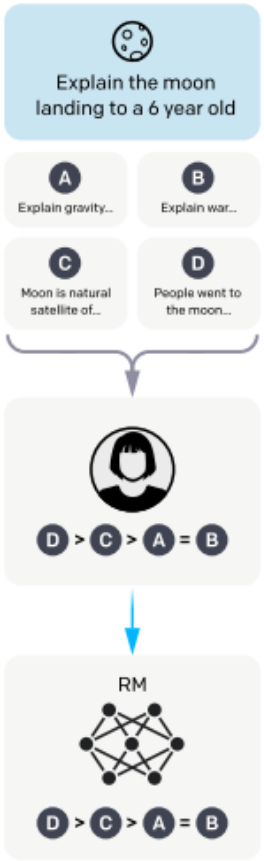

Step 2

**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

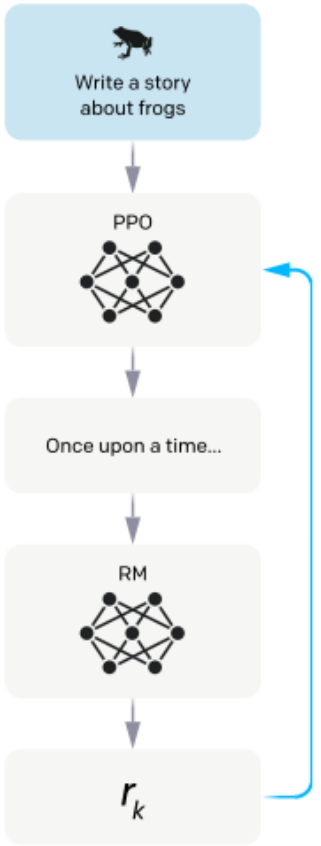
**Optimize a policy against the reward model using reinforcement learning.**

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.



# InstructGPT

## Alignment

## Reinforcement Learning with Human Feedback (RLHF)

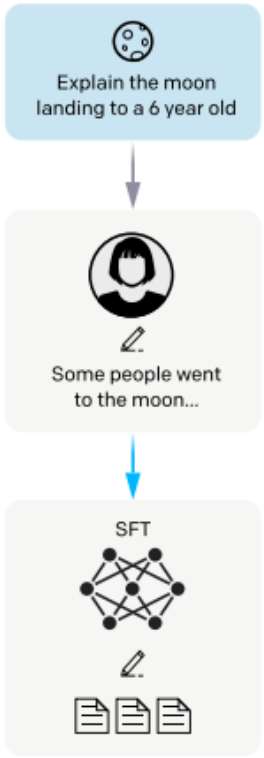
Step 1

**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.



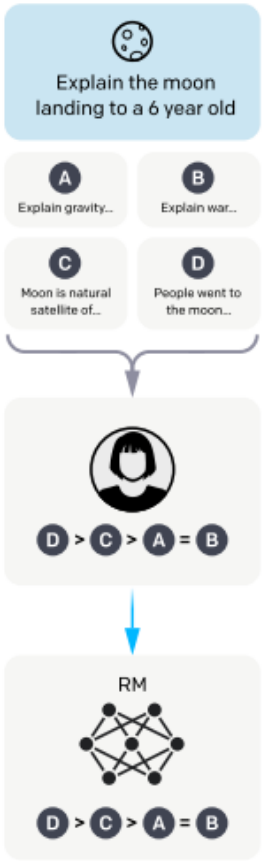

Step 2

**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

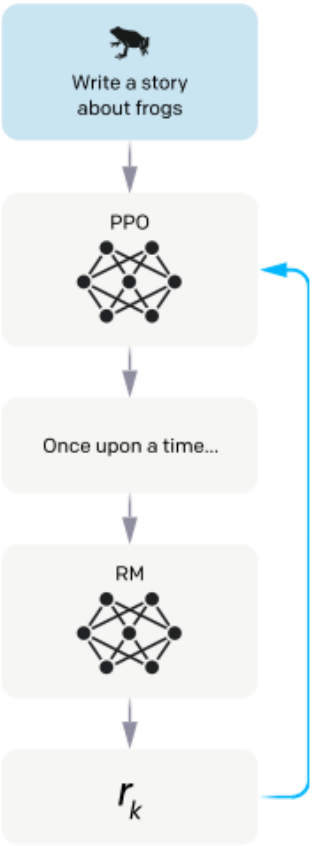
**Optimize a policy against the reward model using reinforcement learning.**

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.



# Zero-Shot CoT Prompting [Kojima et al., NeurIPS 2022]

## Large Language Models are Zero-Shot Reasoners

**Takeshi Kojima**

**Shixiang Shane Gu**
Google Research, Brain Team

**tsuo**
of Tokyo

**Yusuke Iwasawa**
The University of Tokyo

**tract**

s) are widely used in many sub-fields of
generally known as excellent *few-shot*
otably, chain of thought (CoT) prompting,
x multi-step reasoning through step-by-
e-of-the-art performances in arithmetics
*n-2* tasks that do not follow the standard

Large language models are zero-shot reasoners
T Kojima, SS Gu, M Reid, Y Matsuo, Y Iwasawa
Advances in neural information processing systems, 2022 • proceedings.neurips.cc

**Abstract**
Pretrained large language models (LLMs) are widely used in many sub-fields of natural language processing (NLP) and generally known as excellent few-shot learners with task-specific exemplars. Notably, chain of thought (CoT) prompting, a recent technique for eliciting complex multi-step reasoning through step-by-step answer examples, achieved the state-of-the-art performances in arithmetics and symbolic reasoning, difficult system-2 tasks that do not follow the standard scaling laws for LLMs. While these successes are

SHOW MORE

☆ Save 99 Cite Cited by 11003 Related articles All 11 versions »

# "*Let's think step by step.*"

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

---

*(Output) The answer is 8.* ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

---

*(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4.* ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

---

*(Output) 8* ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

---

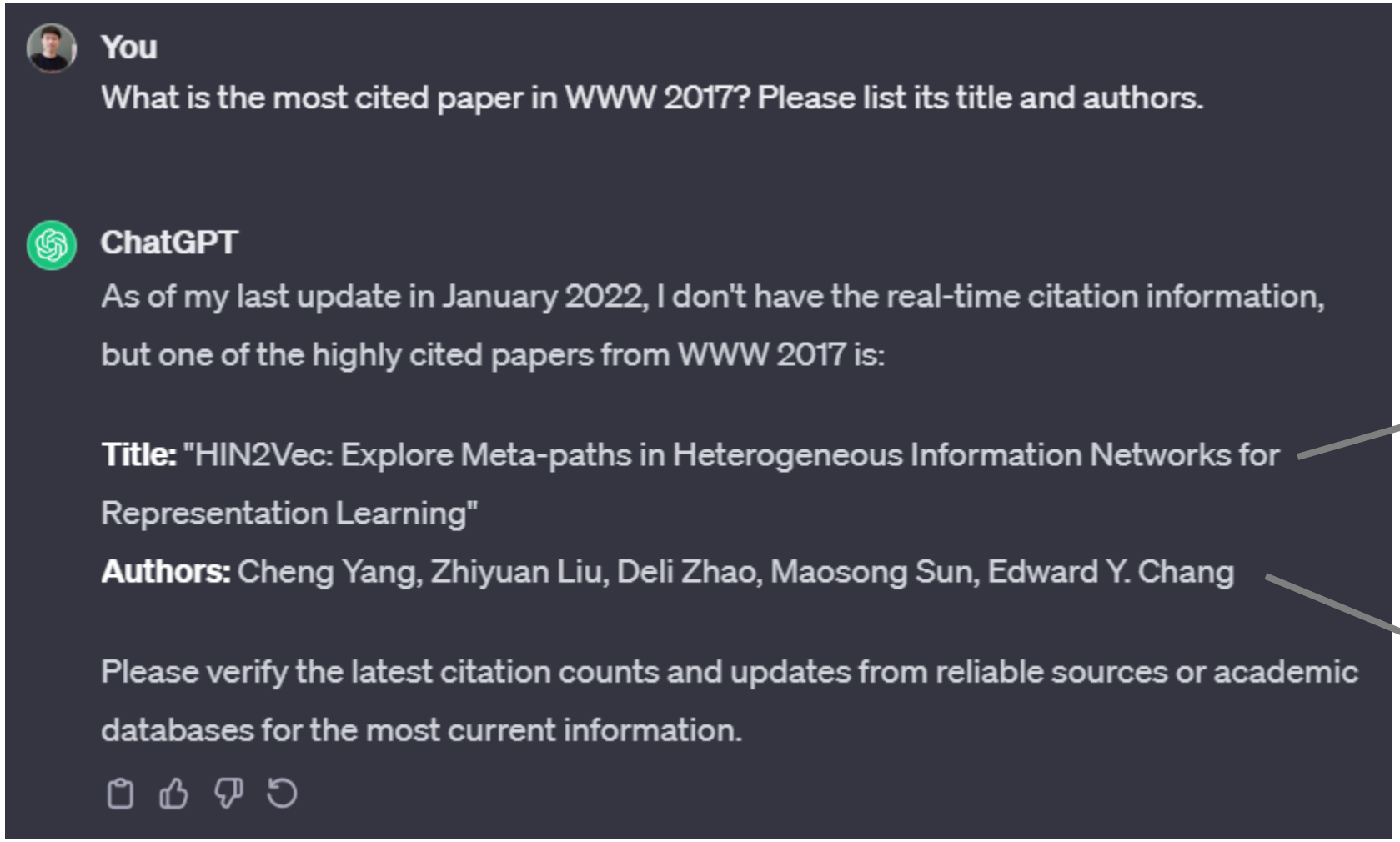
*(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls.* ✓

# The Power of "*Let's think step by step.*"

| No. | Category | Template | Accuracy |
|---|---|---|---|
| 1 | instructive | Let's think step by step. | **78.7** |
| 2 | | First, (*1) | 77.3 |
| 3 | | Let's think about this logically. | 74.5 |
| 4 | | Let's solve this problem by splitting it into steps. (*2) | 72.2 |
| 5 | | Let's be realistic and think step by step. | 70.8 |
| 6 | | Let's think like a detective step by step. | 70.3 |
| 7 | | Let's think | 57.5 |
| 8 | | Before we dive into the answer, | 55.7 |
| 9 | | The answer is after the proof. | 45.7 |
| 10 | misleading | Don't think. Just feel. | 18.8 |
| 11 | | Let's think step by step but reach an incorrect answer. | 18.7 |
| 12 | | Let's count the number of "a" in the question. | 16.7 |
| 13 | | By using the fact that the earth is round, | 9.3 |
| 14 | irrelevant | By the way, I found a good restaurant nearby. | 17.5 |
| 15 | | Abrakadabra! | 15.5 |
| 16 | | It's a beautiful day. | 13.1 |
| - | | (Zero-shot) | 17.7 |

# Large Language Models with Search Engines (Retrieval-Augmented Generation)

# What if an LLM does not have the knowledge to answer a question?




HIN2Vec was published in CIKM 2017.

HIN2Vec was written by Tao-yang Fu, Wang-Chien Lee, and Zhen Lei.

ChatGPT 3.5, queried on January 23, 2024

# What if an LLM does not have the knowledge to answer a question?

- Why does this happen?
  - Knowledge cutoffs: parameters are usually only updated to a particular time
  - Private data: data stored in private text or data repositories not suitable for training
  - Learning failures: even for data that the model was trained on, it might not be sufficient to get the right answer
- How to solve this?
  - Fine-tune the LLM to know one more fact ✗
  - Teach the LLM to explore the external, up-to-date information ✓
  - "*Give a man a fish, and you feed him for a day; teach a man to fish, and you feed him for a lifetime.*"

# Retrieval-Augmented Generation [Lewis et al., NeurIPS 2020]

**Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**

Retrieval-augmented generation for knowledge-intensive nlp tasks

P Lewis, E Perez, A Piktus, F Petroni, V Karpukhin, N Goyal, H Küttler, M Lewis, W Yih...

Advances in neural information processing systems, 2020 • proceedings.neurips.cc

**Abstract**

Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance for their decisions and updating their world knowledge remain open research problems. Pre-trained models with a differentiable

SHOW MORE

☆ Save 99 Cite Cited by 32682 Related articles All 24 versions »

†‡, **Ethan Perez***,

**ir Karpukhin†, Naman Goyal†, Heinrich Küttler†,**

**äschel†‡, Sebastian Riedel†‡, Douwe Kiela†**

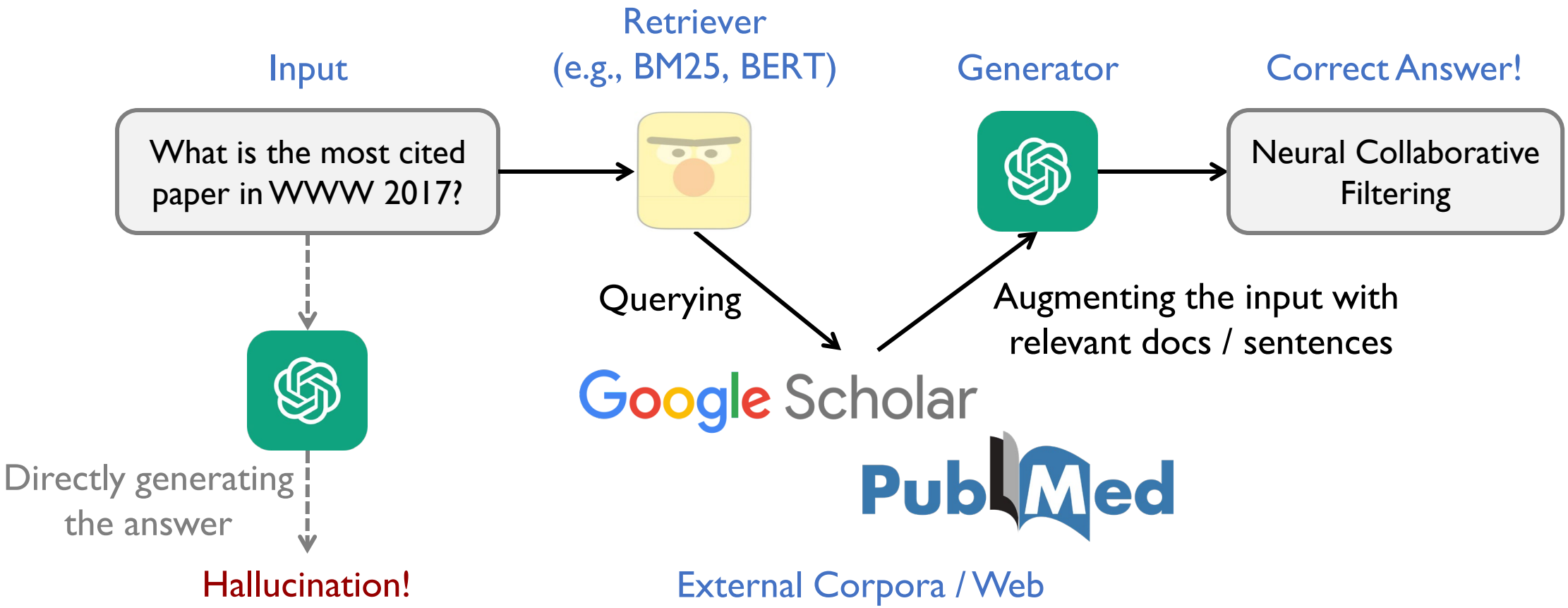
College London; *New York University;
s@fb.com

**stract**

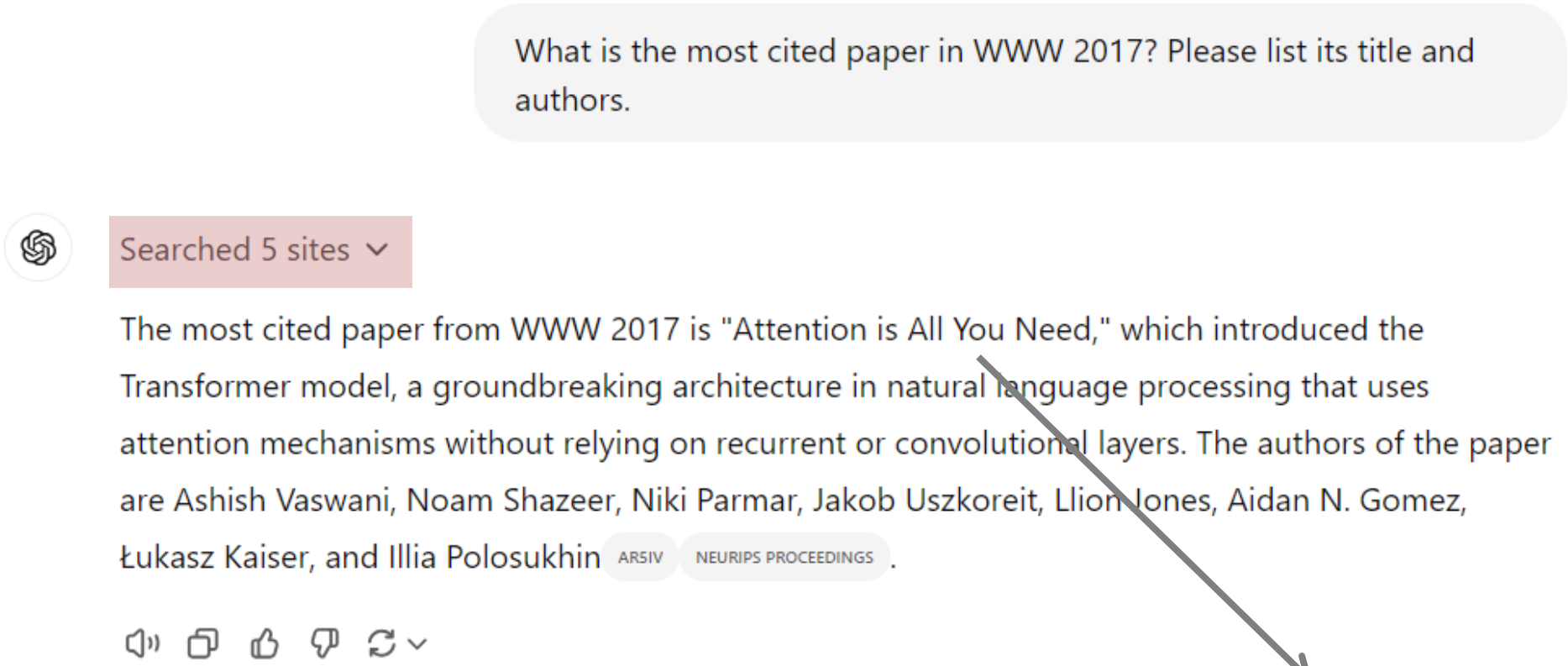
ve been shown to store factual knowledge
f-the-art results when fine-tuned on down-
bility to access and precisely manipulate

# Retrieval-Augmented Generation (RAG)

- Simplest version: single round with a pre-specified plan



# What if single-round retrieval is not enough?




ChatGPT 4o, queried on October 12, 2024

Transformer was published in NeurIPS 2017.

# Toolformer [Schick et al., NeurIPS 2023]

**Toolformer:**
**Language Models Can Teach Themselves to Use Tools**

Toolformer: Language models can teach themselves to use tools

T Schick, J Dwivedi-Yu, R Dessì, R Raileanu, M Lomeli, E Hambro, L Zettlemoyer...

Advances in neural information processing systems, 2023 • proceedings.neurips.cc

**Abstract**

Language models (LMs) exhibit remarkable abilities to solve new tasks from just a few examples or textual instructions, especially at scale. They also, paradoxically, struggle with basic functionality, such as arithmetic or factual lookup, where much simpler and smaller specialized models excel. In this paper, we show that LMs can teach themselves to use external tools via simple APIs and achieve the best of both worlds. We introduce Toolformer, a model trained to decide which APIs to call, when to call them, what

SHOW MORE

☆ Save 99 Cite Cited by 8208 Related articles All 9 versions »

**Roberto Dessì**[†] **Roberta Raileanu**

**tlemoyer Nicola Cancedda Thomas Scialom**

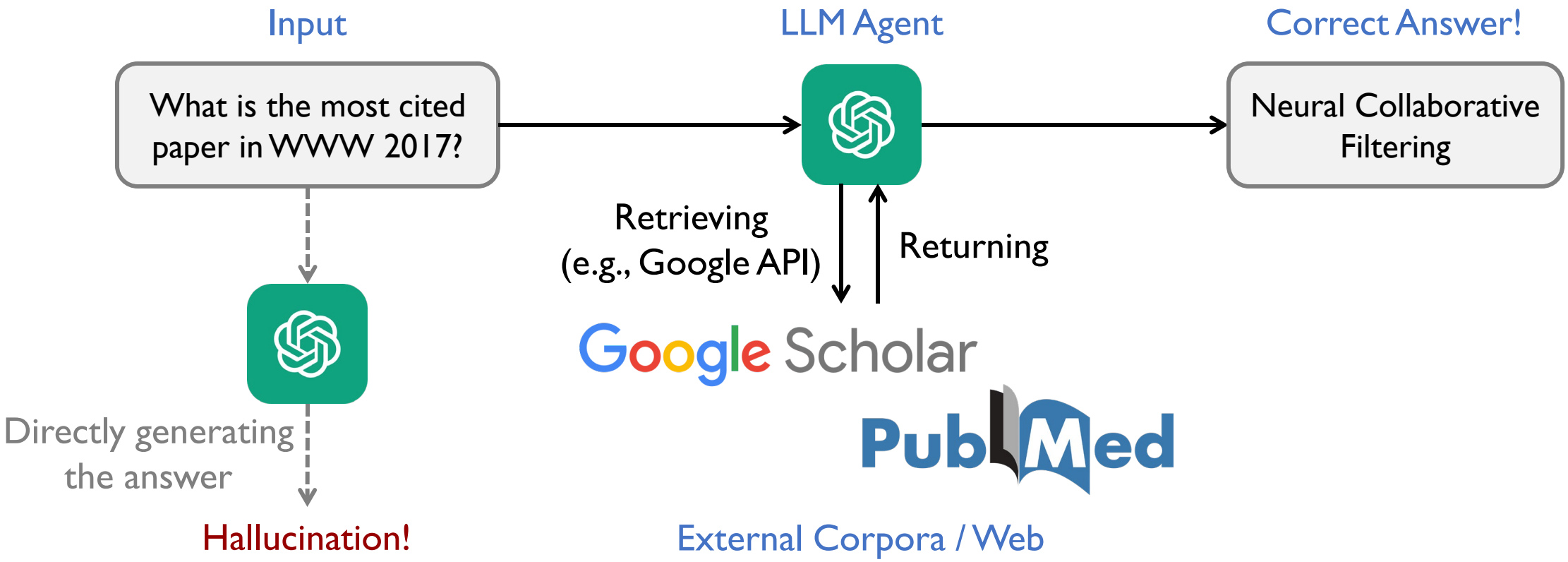
iversitat Pompeu Fabra

**bstract**

rkable abilities to solve new tasks from just a especially at scale. They also, paradoxically, as arithmetic or factual lookup, where much els excel. In this paper, we show that LMs *ools* via simple APIs and achieve the best of *er*, a model trained to decide which APIs to nts to pass, and how to best incorporate the is is done in a self-supervised way, requiring

# Retrieval-Augmented Generation (RAG)

- Advanced version: multi-round, self-adaptive, agentic
- Let the model decide when to search, how many rounds to perform, how to formulate queries, …



# …, and multi-tool

- Retrieval is just one "tool" that can be used by LLMs.
- There are many other "tools".

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.

- Tool use: switching between the text-generation mode and the tool-execution mode

# How to induce tool use?

- Solution 1: Few-shot prompting

*Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "*[QA(question)]*" where "*question*" is the question you want to ask. Here are some examples of API calls:*

**Input:** Joe Biden was born in Scranton, Pennsylvania.
**Output:** Joe Biden was born in **[QA("Where was Joe Biden born?")]** Scranton, **[QA("In which state is Scranton?")]** Pennsylvania.

**Input:** Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.
**Output:** Coca-Cola, or **[QA("What other name is Coca-Cola known by?")]** Coke, is a carbonated soft drink manufactured by **[QA("Who manufactures Coca-Cola?")]** the Coca-Cola Company.

**Input: x**
**Output:**

- Solution 2: Supervised fine-tuning (if you have 1,000 training samples of how to use a tool, you can fine-tune the model)

# Solution 1: Few-Shot Prompting

- Inference-time technique
- Put reasoning and action examples in the prompt, then ask the question



question



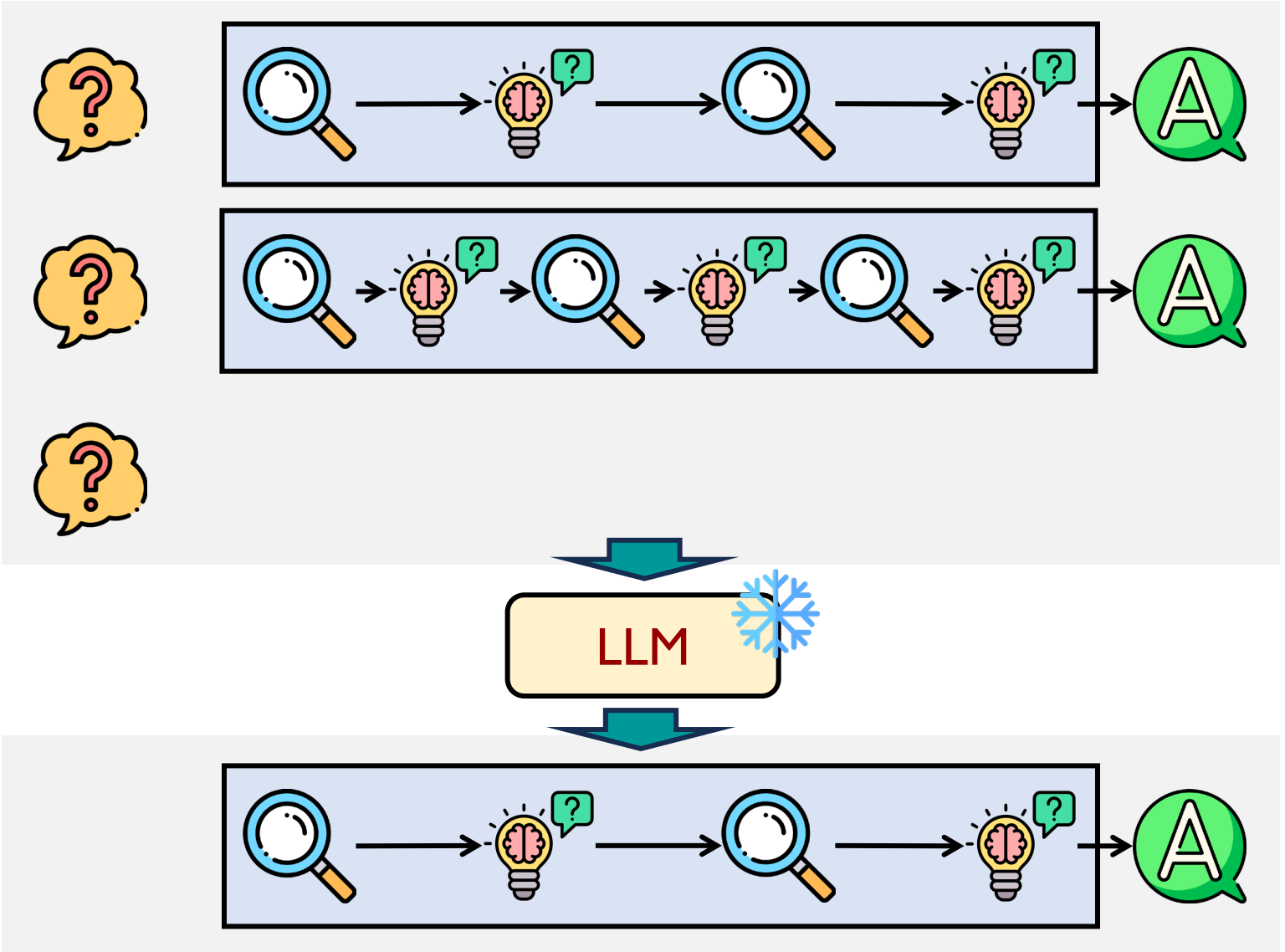
search
(tool execution)



reasoning
(text generation)

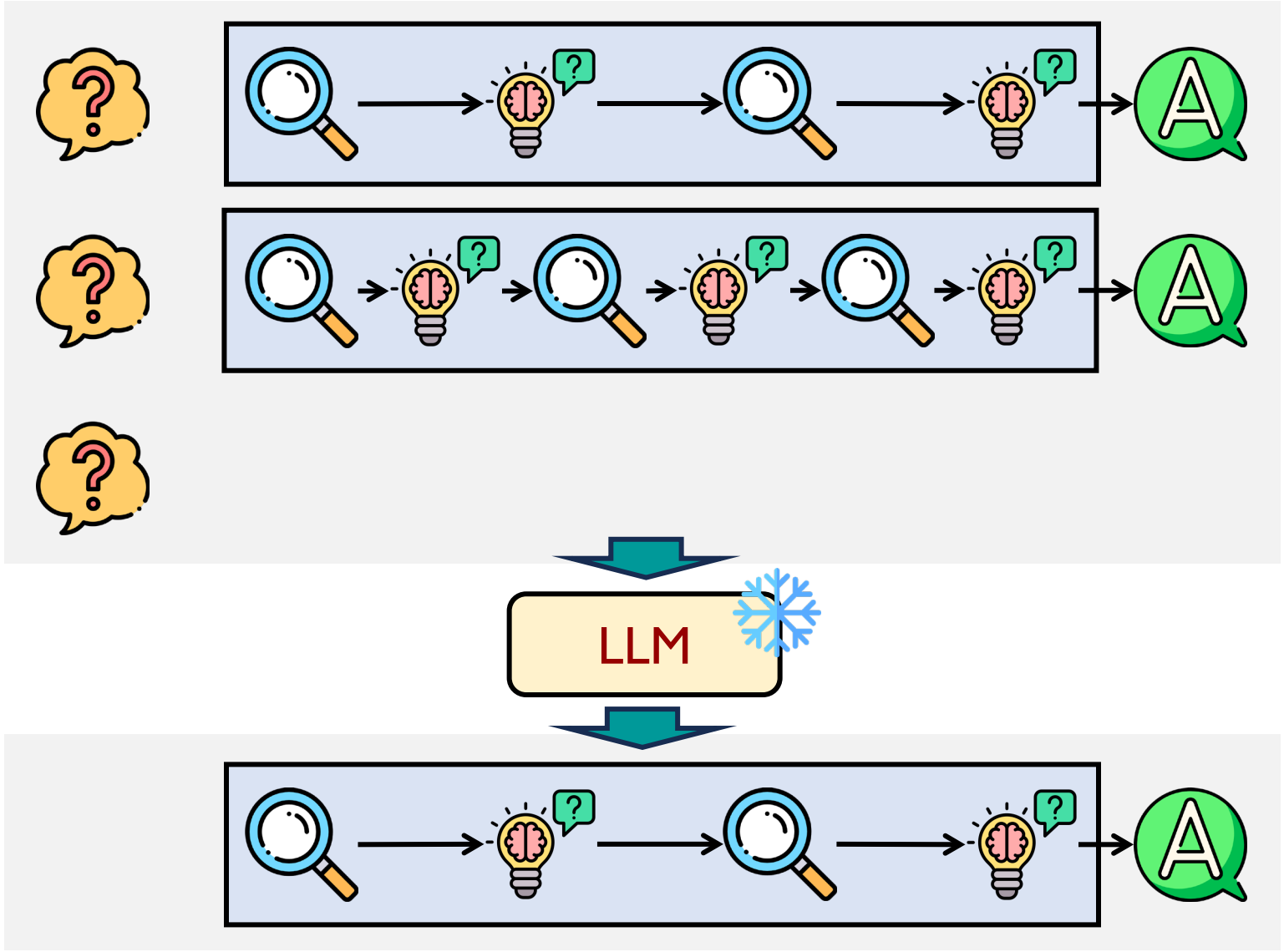


answer



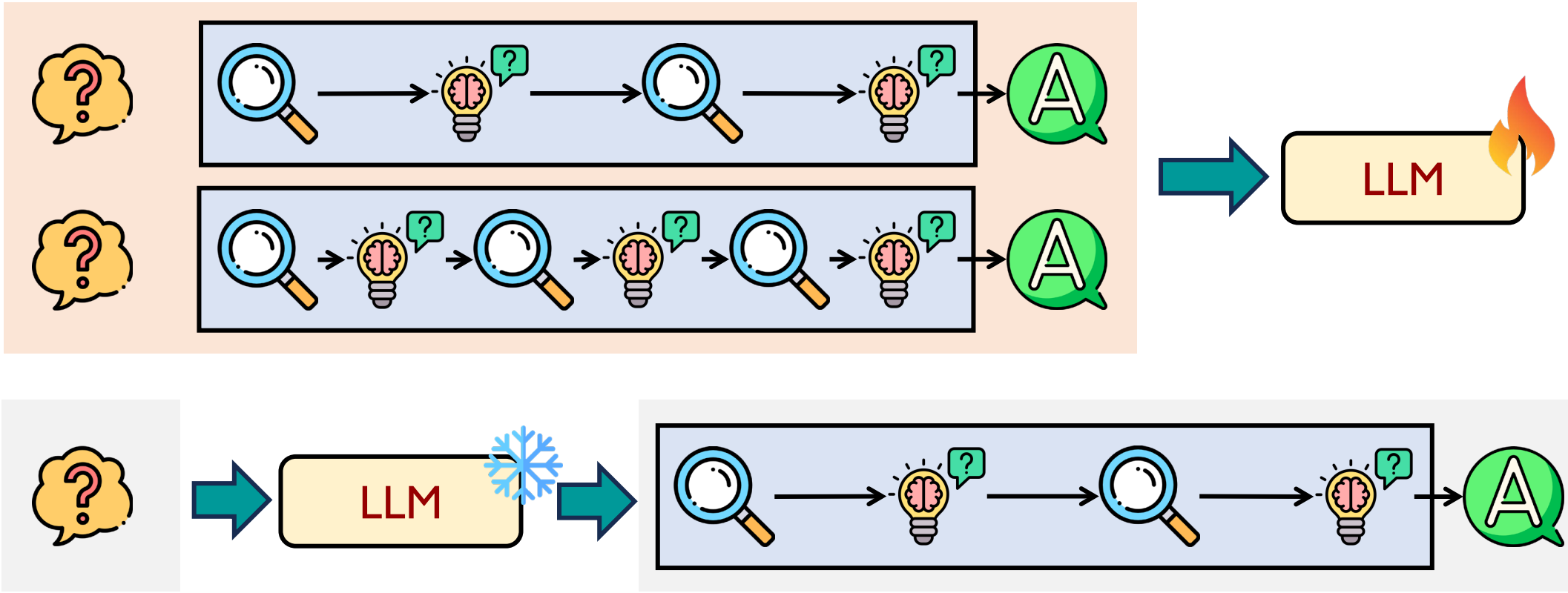
# Solution 1: Few-Shot Prompting

- Limitation: The ability to call a search engine is not internalized through model parameter updates



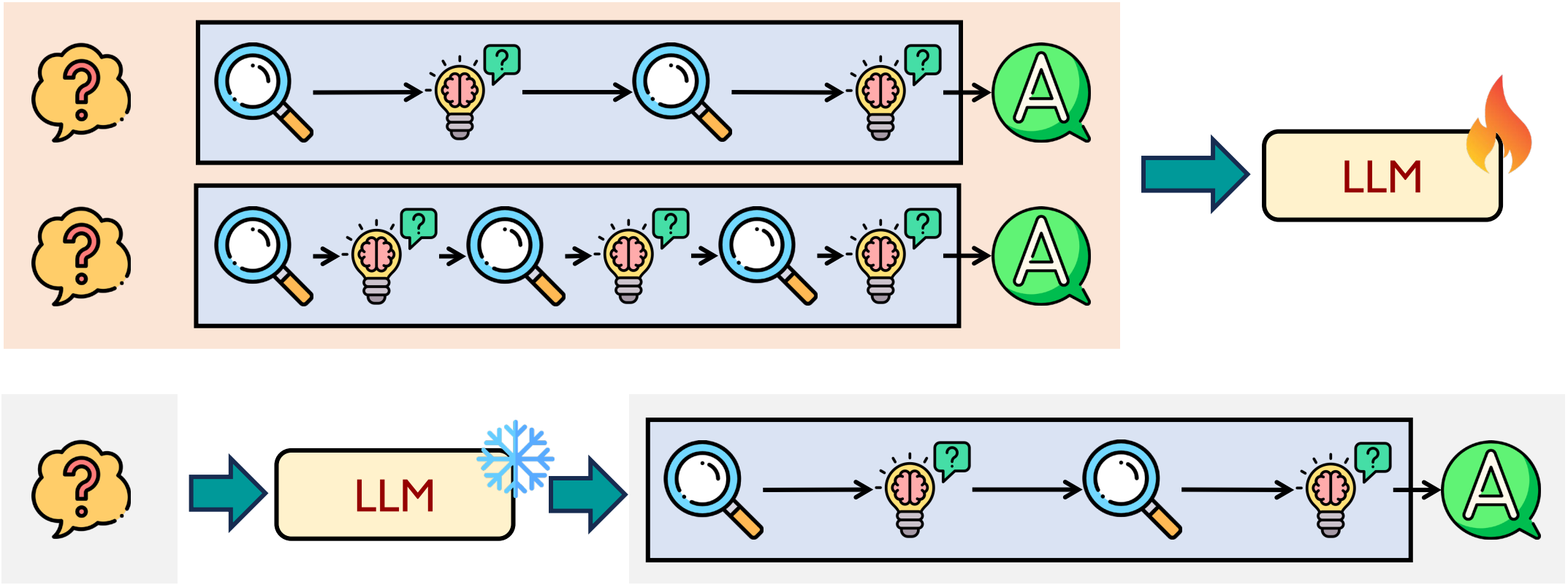
# Solution 2: Supervised Fine-Tuning

- Training-based solution
- Collect reasoning and search trajectories to fine-tune the LLM



# Solution 2: Supervised Fine-Tuning

- Limitation 1: Hard to scale due to reliance on high-quality labeled trajectories
- Limitation 2: Poor generalization to unseen tools (e.g., trained on BM25 but deployed on Google API)



# Training LLMs with Reinforcement Learning

# DeepSeek-R1: Training LLMs to Reason with RL

- The correct answer (finally…):




DeepSeek-R1 671B, queried on May 8, 2025

# Scratchpad of DeepSeek-R1

Reasoning
Started thinking...

Okay, so I need to find out the most cited paper from the WWW 2017 conference. Let me start by recalling that WWW stands for the International World Wide Web Conference, which is a major venue for research related to the web. The 2017 edition would have been the 26th conference.

First, I should figure out how to determine the most cited paper from that specific year. Citation counts can be tracked through databases like Google Scholar, Semantic Scholar, or the ACM Digital Library since ACM often sponsors these conferences. However, citation counts can vary over time, so the most cited paper might change as newer papers accumulate citations. But since the user is asking about the current most cited, I need up-to-date information.

I'll start by checking the proceedings of WWW 2017. The conference usually publishes its proceedings, and sometimes the awards or notable papers are mentioned. If I can find the list of accepted papers, I can then look up each of them on Google Scholar to check their citation counts. But that's time-consuming. Alternatively, maybe there's a known paper from that year that's highly cited. For example, in recent years, transformer-based models like BERT or attention mechanisms have been highly cited, but 2017 might be a bit early for those.

Wait, 2017 is the year when the "Attention Is All You Need" paper was published, but that was at NIPS, not WWW. So maybe that's not relevant here. Let me think. In the realm of web conferences, topics like graph neural networks, recommendation systems, network analysis, and so on are common.

Another approach is to search for "most cited papers from WWW 2017" on Google or academic search engines. Let me try that. Searching for that exact phrase, I find some resources. For instance, there's a page on the WWW 2017 conference website that lists accepted papers, but I need to check if they have any awards or most cited mentions. Alternatively, Semantic Scholar might have a list sorted by citations.

Let me check Semantic Scholar. Going to their website, I can search for papers from the WWW 2017 conference. Applying the filters for the year 2017 and the venue WWW, then sorting by citations. Let's see. The top result there might be the one. Alternatively, Google Scholar can be used by searching for the conference name and year, then sorting by citations.

Upon doing a quick search, I find that the paper "Inductive Representation Learning on Large Graphs" by William L. Hamilton, Rex Ying, and Jure Leskovec is highly cited. Wait, but that was published at NIPS 2017, right? Wait, no, maybe I'm confused. Let me check the details. The Hamilton et al. paper on GraphSAGE was indeed presented at NIPS 2017. So maybe that's not the one.

Alternatively, another paper from WWW 2017 that's highly cited could be related to graph convolutional networks or something similar. Let me check the proceedings. For example, "Graph Convolutional Matrix Completion" by Rianne van den Berg, Thomas N. Kipf, and Max Welling was published in 2017, but I think that's at a different conference.

Another possibility is "Neural Collaborative Filtering" by Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Wait, but that paper was published in WWW 2017? Let me confirm. According to Google Scholar, that paper was indeed published in the Proceedings of the 26th International Conference on World Wide Web in 2017. It's possible that this paper is highly cited as it introduces neural networks into collaborative filtering, a popular approach in recommendations.

Checking the citation counts: As of now, "Neural Collaborative Filtering" has over 5,000 citations on Google Scholar, which is very high. Another paper, "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction" has over 2,000 citations. There might be others. Wait, but I need to confirm which one has the highest.

Alternatively, "Inductive Representation Learning on Large Graphs" (GraphSAGE) by Hamilton et al. has over 6,000 citations, but that's from NIPS 2017, not WWW. So perhaps the most cited from WWW 2017 is "Neural Collaborative Filtering". Let me cross-verify.

Looking at the WWW 2017 accepted papers list, "Neural Collaborative Filtering" is indeed there. The authors are Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. That paper is foundational in combining neural networks with collaborative filtering, which explains its high citation count.

Alternatively, another paper from WWW 2017 that's highly cited is "StarSpace: Embed All The Things!" by Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. This paper also has a substantial number of citations, but I think it's less than "Neural Collaborative Filtering".

To be sure, checking Google Scholar for "Neural Collaborative Filtering" shows over 5,000 citations as of 2023. "DeepFM" has around 2,000. Another paper, "Metapath2vec: Scalable Representation Learning for Heterogeneous Networks" by Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami, which was also presented at WWW 2017, has over 1,500 citations.

Therefore, it's likely that "Neural Collaborative Filtering" by He et al. is the most cited paper from WWW 2017. However, I should also check if there's any other paper with higher citations. For example, "Attention Is All You Need" is from 2017 but not WWW. "Dynamic Routing Between Capsules" by Hinton et al. is also from 2017 but a different conference.

Another possible candidate is "Semi-Supervised Classification with Graph Convolutional Networks" by Kipf and Welling, but that was published at ICLR 2017, so not WWW.

Therefore, after checking the citation counts and cross-referencing with the conference proceedings, "Neural Collaborative Filtering" by Xiangnan He and colleagues seems to be the most cited paper from WWW 2017. However, I should verify once more. Let me check another source.

On Semantic Scholar, searching for WWW 2017 papers sorted by citations: the top result is "Neural Collaborative Filtering" with 5,442 citations as of October 2023. The next is "Metapath2vec" with 2,035 citations. Then "DeepFM" with 1,821. So yes, "Neural Collaborative Filtering" is the most cited.

Thus, the answer should be that paper.

# Scratchpad of DeepSeek-R1

Another approach is to search for "most cited papers from WWW 2017" on Google or academic search engines. Let me try that. Searching for that exact phrase, I find some resources. For instance, there's a page on the WWW 2017 conference website that lists accepted papers, but I need to check if they have any awards or most cited mentions. Alternatively, Semantic Scholar might have a list sorted by citations.

Let me check Semantic Scholar. Going to their website, I can search for papers from the WWW 2017 conference. Applying the filters for the year 2017 and the venue WWW, then sorting by citations. Let's see. The top result there might be the one. Alternatively, Google Scholar can be used by searching for the conference name and year, then sorting by citations.

Checking the citation counts: As of now, "Neural Collaborative Filtering" has over 5,000 citations on Google Scholar, which is very high. Another paper, "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction" has over 2,000 citations. There might be others. Wait, but I need to confirm which one has the highest.

Alternatively, "Inductive Representation Learning on Large Graphs" (GraphSAGE) by Hamilton et al. has over 6,000 citations, but that's from NIPS 2017, not WWW. So perhaps the most cited from WWW 2017 is "Neural Collaborative Filtering". Let me cross-verify.

# DeepSeek-R1 [Guo et al., Nature 2025]

Article | Open access | Published: 17 September 2025

## DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Bei Feng, Chengda Lu, ... Zhen

Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning

D Guo, D Yang, H Zhang, J Song, P Wang, Q Zhu, R Xu, R Zhang, S Ma, X Bi, X Zhang, X Yu...

arXiv preprint arXiv:2501.12948, 2025 • arxiv.org

General reasoning represents a long-standing and formidable challenge in artificial intelligence. Recent breakthroughs, exemplified by large language models (LLMs) and chain-of-thought prompting, have achieved considerable success on foundational reasoning tasks. However, this success is heavily contingent upon extensive human-annotated demonstrations, and models' capabilities are still insufficient for more complex problems. Here we show that the reasoning abilities of LLMs can be incentivized through

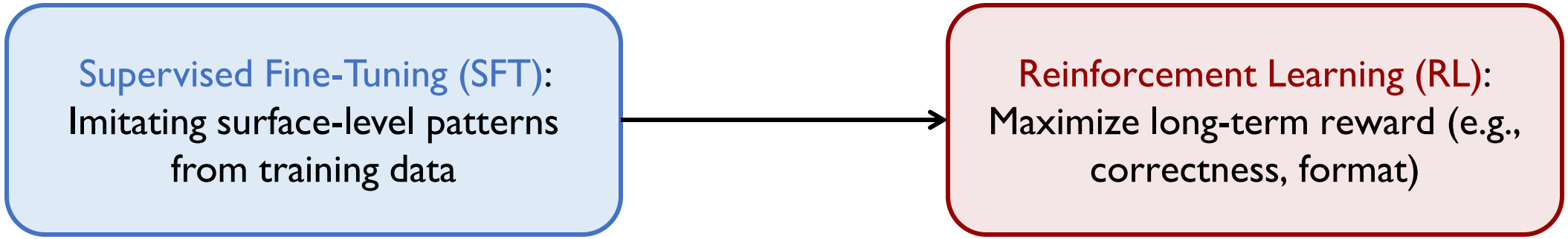
SHOW MORE

☆ Save 99 Cite Cited by 14055 Related articles All 2 versions »

formidable challenge in artificial

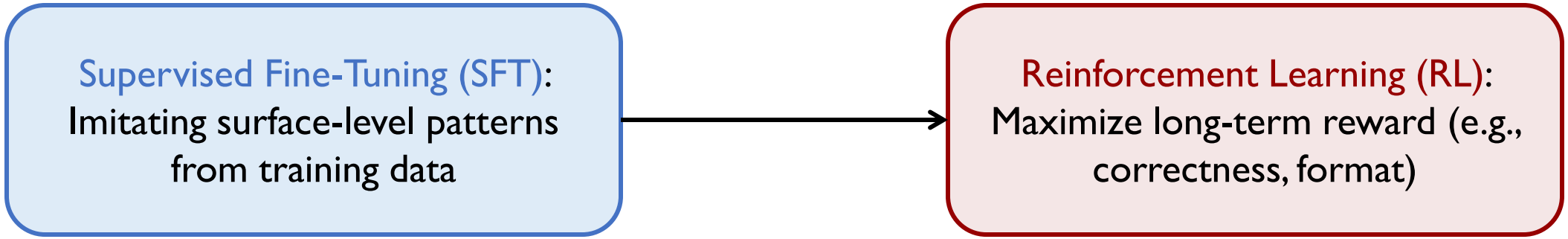
ed by large language models (LLMs)[1,2] and

# DeepSeek-R1: Training LLMs to Reason with RL



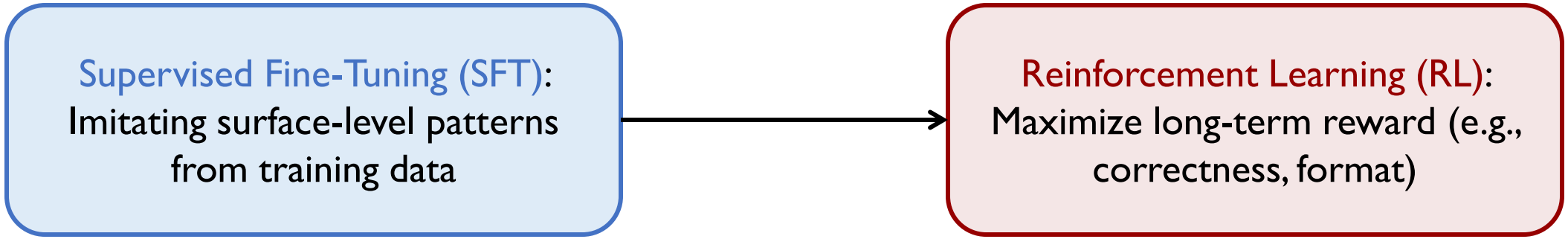

- Supervised Fine-Tuning (SFT): Given a question, maximize the likelihood that the LLM generates the reference answer.
  - If the reference answer is 0.5, then generating 1/2 is wrong.
  - If the LLM is asked to write code for a specific function, only an exact match with the reference answer is considered fully correct. A program that differs from the reference by just one token (even if it does not run at all) is regarded as better than an alternative implementation that correctly achieves the same functionality.

# DeepSeek-R1: Training LLMs to Reason with RL




- Reinforcement Learning with Verifiable Rewards (RLVR): When the LLM does something correct according to our rubrics, it gets certain reward. The model is trained to maximize the reward.
  - Format Reward:
    - Has <think> and </think> tokens in its output; puts its intermediate reasoning steps between these two tokens.
    - Has <answer> and </answer> tokens in its output; puts its final answer between these tokens
    - …

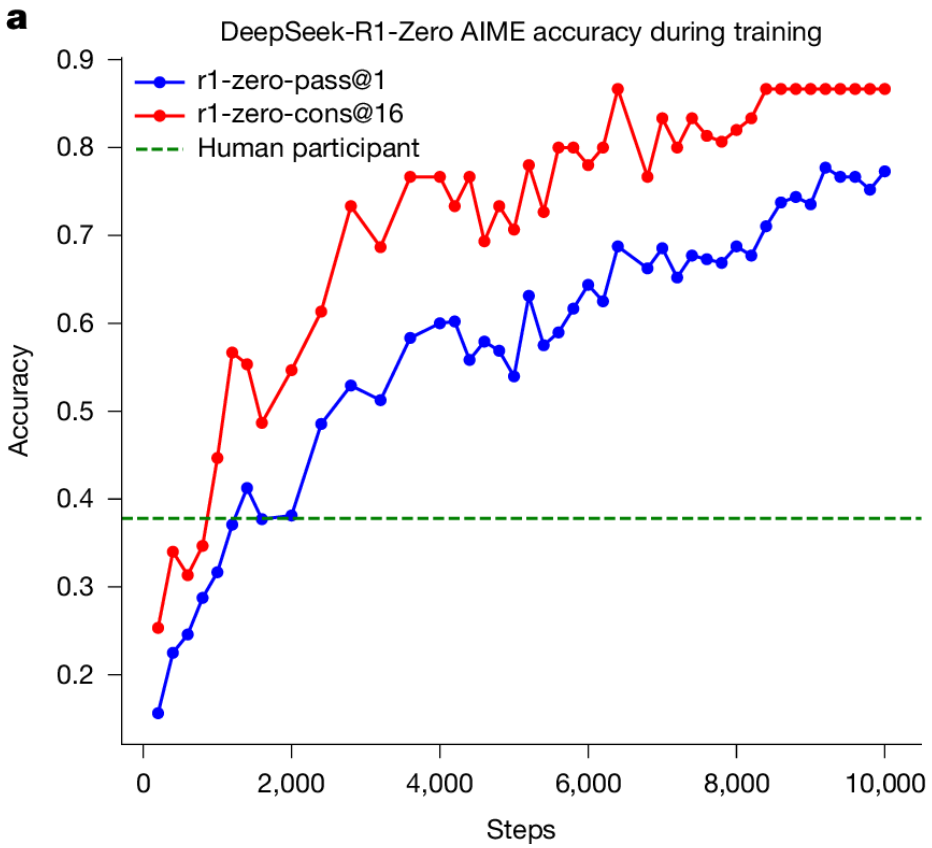
# DeepSeek-R1: Training LLMs to Reason with RL

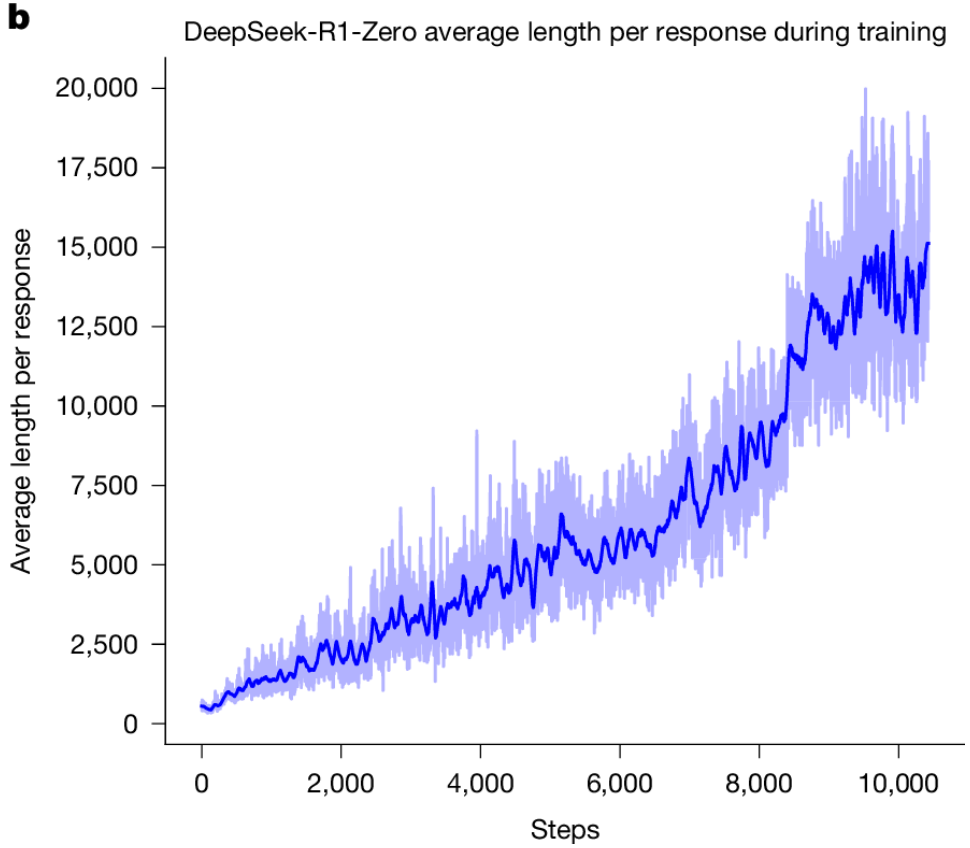



- Reinforcement Learning with Verifiable Rewards (RLVR): When the LLM does something correct according to our rubrics, it gets certain reward. The model is trained to maximize the reward.
  - Correctness Reward:
    - The answer (i.e., tokens between <answer> and </answer>) should be "equivalent" to the reference answer (according to a verifier).
  - If the model does not generate the correct answer, it will NOT know the answer after this iteration; it will only know that the reward is low.
    - This question can be used repeatedly until the model generates a good answer.

# DeepSeek-R1: Training LLMs to Reason with RL

- When the model can repeatedly try while only receiving binary feedback (correct/incorrect), it is forced to engage in longer reasoning before reaching the correct answer.
- Experiments show this is effective.

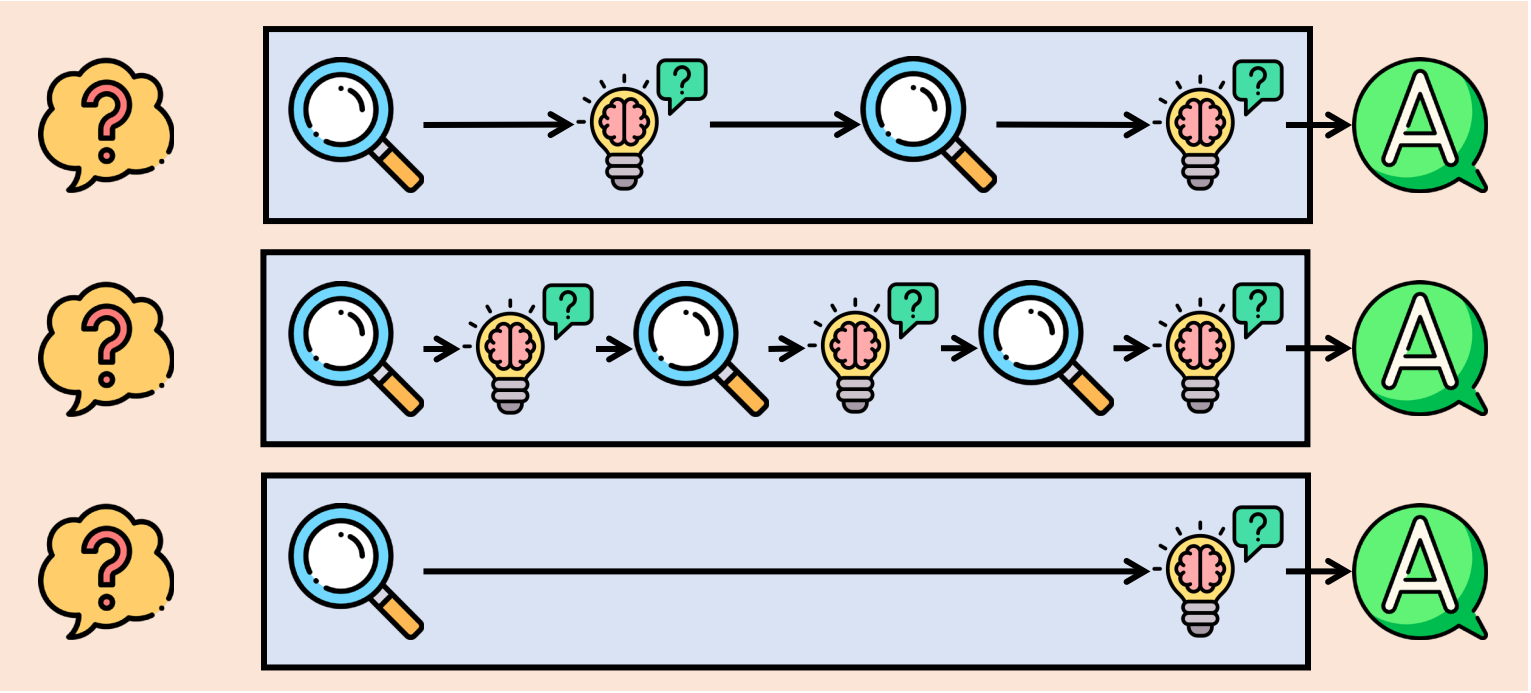


# Training LLMs to Search with RL?

- Motivation:
  - Because DeepSeek-R1 is popular ✗
  - Because we no longer need annotated trajectories! ✓

I do not have this, …



…, but I have this
(which is enough for RL)!



# Next Lecture: Search-R1 [Jin et al., COLM 2025]

## Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning

**Bowen Jin**[1], **Hansi Zeng**[2], **Zhenrui Yue**[1], **Jinsung Yoon**[3], **Sercan Ö. Arık**[3], **Dong Wang**[1], **Hamed Zamani**[2], **Jiawei Han**[1]
[1] Department of Computer Science, University of Illinois at Urbana-Champaign
[2] Center for Intelligent Information Retrieval, University of Massachusetts Amherst
[3] Google Cloud AI Research
{bowenj4,zhenrui3,dwang24,hanj}@illinois.edu, {hzeng, zamani}@cs.umass.edu
{jinsungyoon,soarik}@google.com

### Abstract

Efficiently acquiring external knowledge and up-to-date information is essential for effective reasoning and text generation in large language models (LLMs). Prompting advanced LLMs with reasoning capabili-



# Thank You!

Course Website: https://yuzhang-teaching.github.io/CSCE670-F26.html