香港城市大學（東莞）
City University of Hong Kong
(Dongguan)

Celebrating the Establishment in Year 2024

Lecture 9: LLM Agents

CS6493 Natural Language Processing

Instructor: Linqi Song

Outline

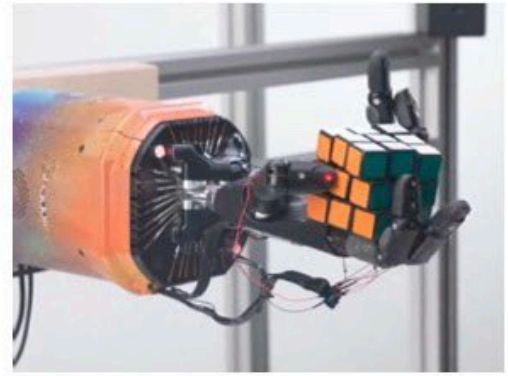
- LLM agents introduction
- Reasoning
- Tool learning
- Knowledge incorporation-RAG

Language models are powerful, but they still suffer from

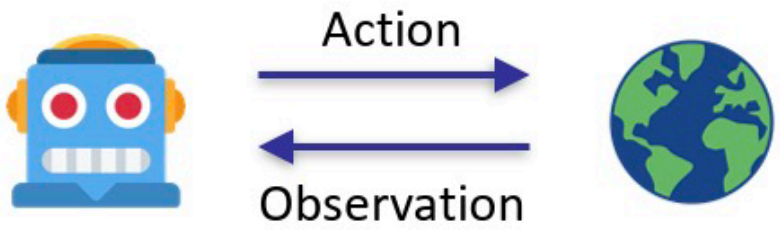
- Hallucination
- Inaccuracy of knowledge
- Lack of interpretability
- Inconsistency
- Limited scalability
- Restricted capabilities
- …

Can LLMs utilize internal reasoning/planning and external tools/knowledge to not only **expand** their **capacities** but also to make our NLP systems more **robust**, **scalable**, and **interpretable**?

Agent examples

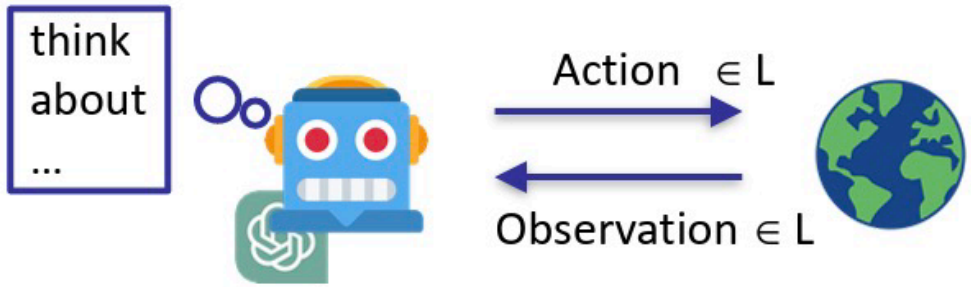


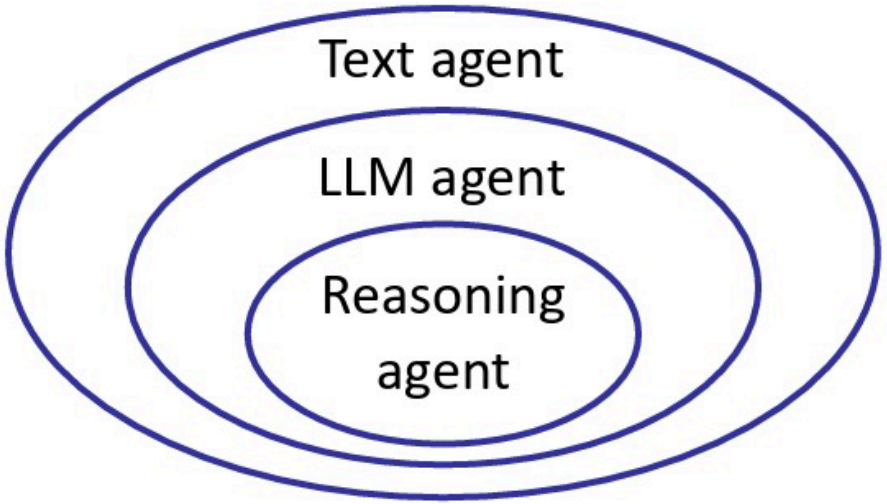
What are agents?



- An “intelligent” system that interacts with some “environment”
  - Physical environments: robot, autonomous car, …
  - Digital environments: DQN for Atari, Siri, AlphaGo, …
  - Humans as environments: chatbot
- Define “agent” by defining “intelligent” and “environment”
  - It changes over time!
  - Exercise question: how would you define “intelligent”?

What are LLM agents?





- Level 1: Text agent
  - Uses text action and observation
  - Examples: ELIZA, LSTM-DQN
- Level 2: LLM agent
  - Uses LLM to act
  - Examples: SayCan, Language Planner
- Level 3: Reasoning agent
  - Uses LLM to reason to act
  - Examples: ReAct, AutoGPT
  - **The key focus of the field and the talk**

Example: SayCan

How PaLM-SayCan Works

This demo shows a PaLM-enabled helper robot performing a series of complex tasks using chain of thought prompting and the step-by-step solution needed to carry out the requests.

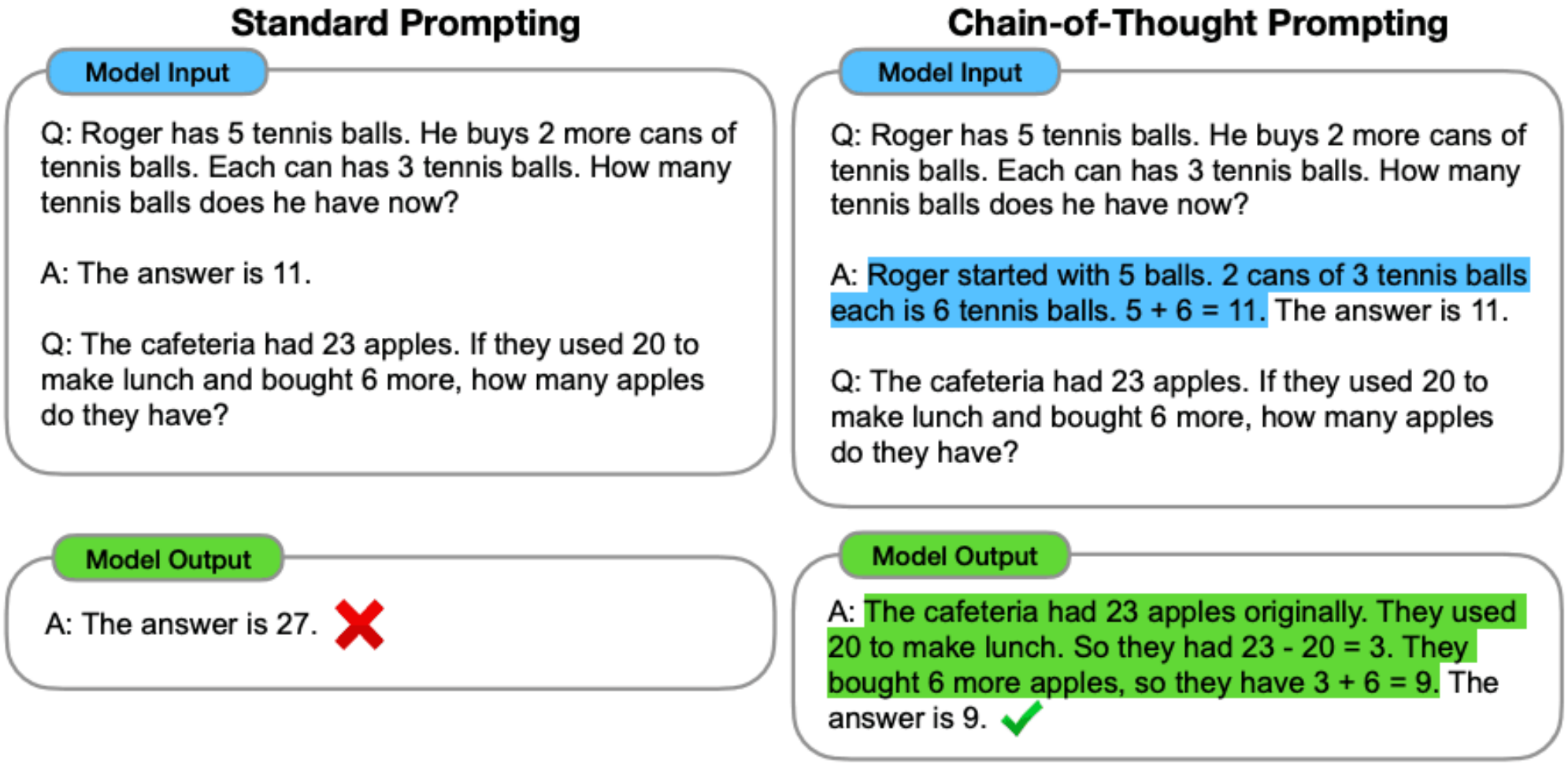
1. Select a task for the helper robot
2. See how PaLM-SayCan interprets the task
3. See how the helper robot executes the task
4. Learn the step-by-step solution behind PaLM-SayCan

I just worked out. Can you bring me a drink and a snack to recover?

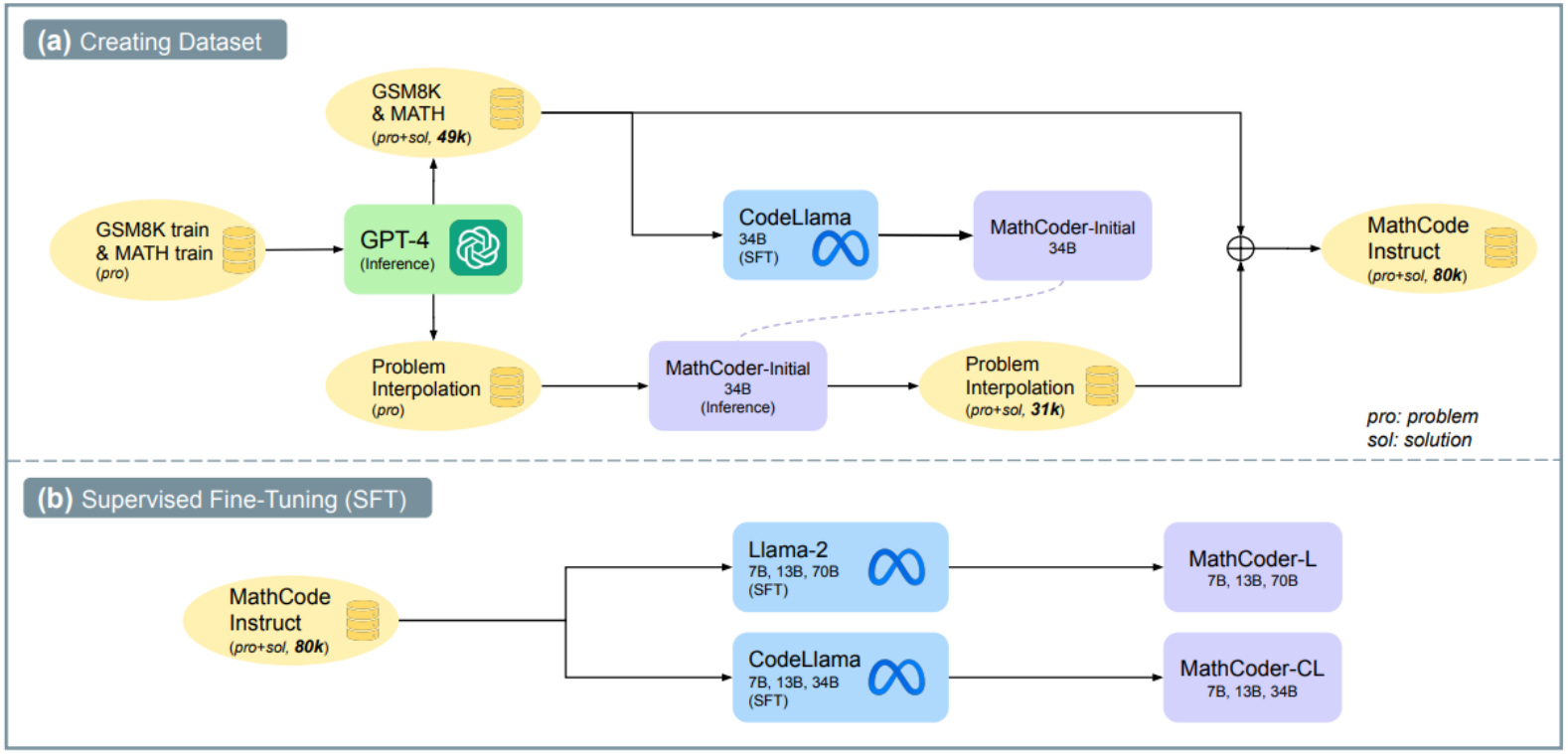
After a task has been requested, PaLM-SayCan uses chain of thought prompting, which interprets the instruction in order to score the likelihood that an individual skill makes progress towards completing the high-level request.

The user has asked for a drink and a snack. I will bring a water bottle and an apple.

LLM reasoning: recall of CoT



LLM reasoning: solving math problems



GSM8K: 8.5K high quality linguistically diverse grade school math word problems

```
Weng earns $12 an hour for
babysitting. Yesterday, she just did
50 minutes of babysitting. How much
did she earn?
```

```
Weng earns 12/60 = $<<12/60=0.2>>0.2 per
minute.
Working 50 minutes, she earned 0.2 x 50 =
$<<0.2*50=10>>10.
#### 10
```

MATH: 12,500 challenging competition mathematics problems

MATH Dataset (Ours)

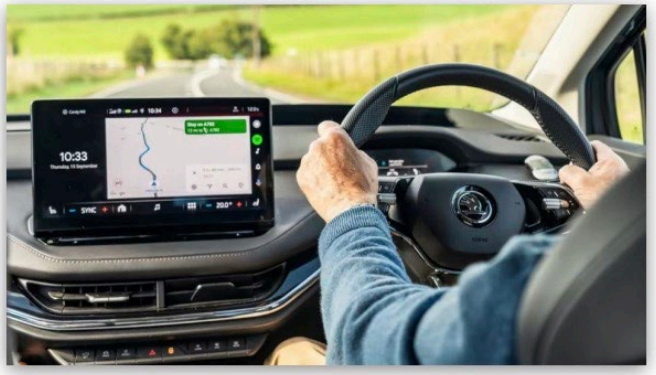
Problem: Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?
Solution: There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.

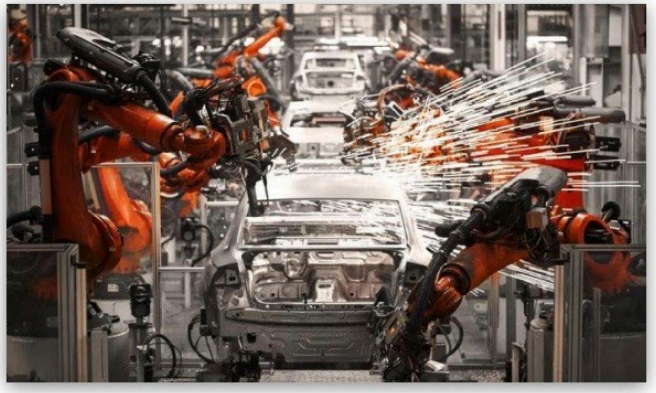
Problem: If $\sum_{n=0}^{\infty} \cos^{2n}\theta = 5$, what is $\cos 2\theta$?
Solution: This geometric series is $1 + \cos^2\theta + \cos^4\theta + \cdots = \frac{1}{1-\cos^2\theta} = 5$. Hence, $\cos^2\theta = \frac{4}{5}$. Then $\cos 2\theta = 2\cos^2\theta - 1 = \boxed{\frac{3}{5}}$.

Problem: The equation $x^2 + 2x = i$ has two complex solutions. Determine the product of their real parts.
Solution: Complete the square by adding 1 to each side. Then $(x+1)^2 = 1 + i = e^{\frac{i\pi}{4}}\sqrt{2}$, so $x + 1 = \pm e^{\frac{i\pi}{8}}\sqrt[4]{2}$. The desired product is then $\left(-1 + \cos\left(\frac{\pi}{8}\right)\sqrt[4]{2}\right)\left(-1 - \cos\left(\frac{\pi}{8}\right)\sqrt[4]{2}\right) = 1 - \cos^2\left(\frac{\pi}{8}\right)\sqrt{2} = 1 - \frac{\left(1+\cos\left(\frac{\pi}{4}\right)\right)}{2}\sqrt{2} = \boxed{\frac{1-\sqrt{2}}{2}}$.

Human + tool use: motivations (1)

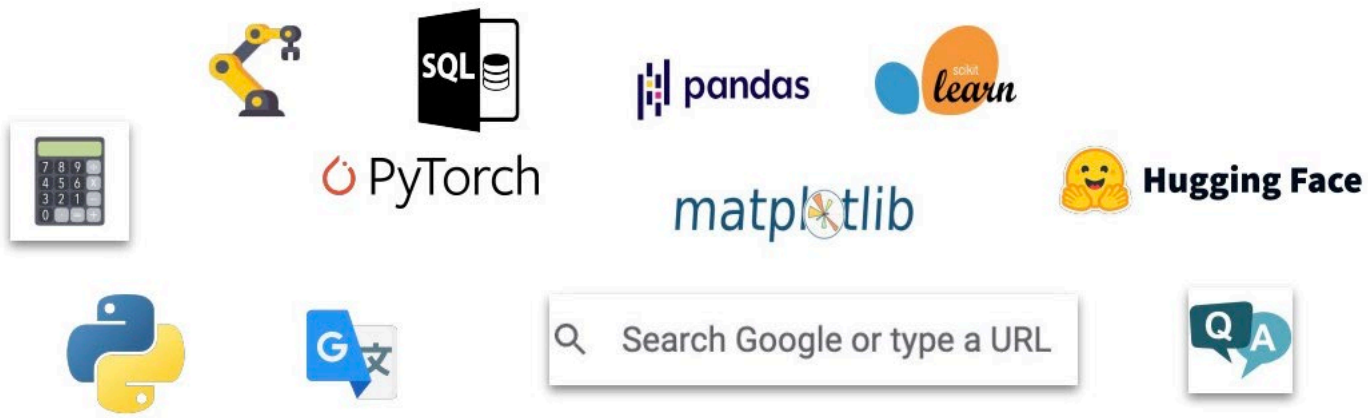
- As humans, we have limited time and memory, feel tired, and have emotions.
- Human + tool use
  - Enhanced scalability
  - Improved consistency
  - Greater interpretability
  - Higher capacity and productivity





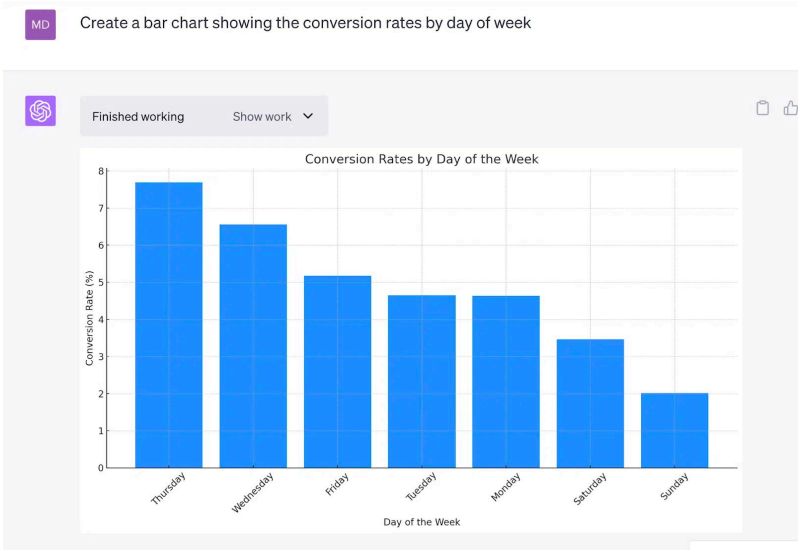
LLMs + tool use: motivations (2)

- Just like humans, LLMs suffer from the similar limitations. But in the same way,
- LLMs + tool use
  - Enhanced scalability
  - Improved consistency
  - Greater interpretability
  - Higher capacity, productivity



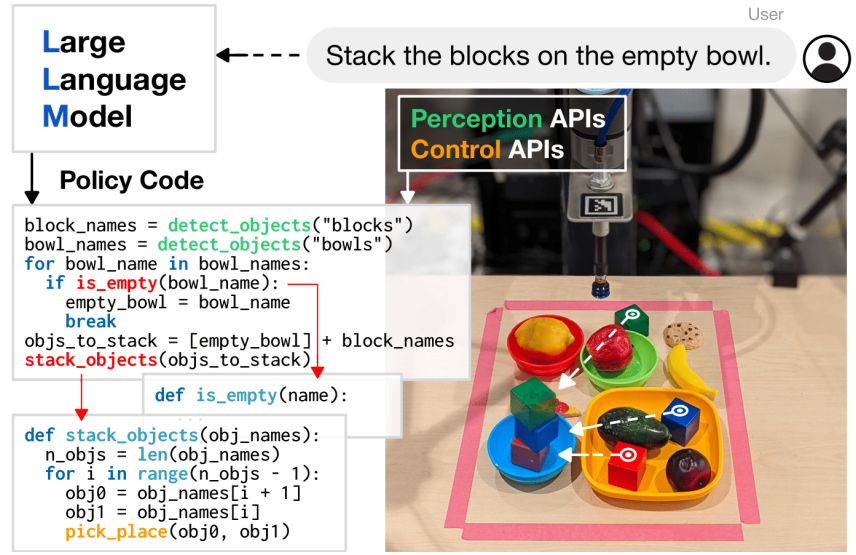
LLMs + tool use in perspective of executable language grounding (1)

- Ground language models into executable actions
- Mapping natural language instructions into code or actions executable within various environments such as databases, web applications, and robotic physical world.
- LM (planning and reasoning) + actions



Data analysis

Web/Apps



Robotic physical world

LLMs + tool use in perspective of executable language grounding (2)

- LLMs + tool use in executable language grounding tasks

Inputs

- Language: user question/request
- Toolkit: code, APIs to search engines, self-defined functions, expert models…
- Environment: databases, IDE, web/apps, visual and robotic physical world…

Outputs

- Grounded reasoning code/action sequence that can be executed in the corresponding environment
  - What tools to select, when and how to use the selected tools

Example of LLMs + tool use in executable language grounding

- LLMs + tool use in executable language grounding
- Language: user question/statement about a database
- Toolkit: code, APIs to NLP functionalities (expert models)
- Environment: databases, SQL/Python IDEs

Binding Language Models in Symbolic Languages

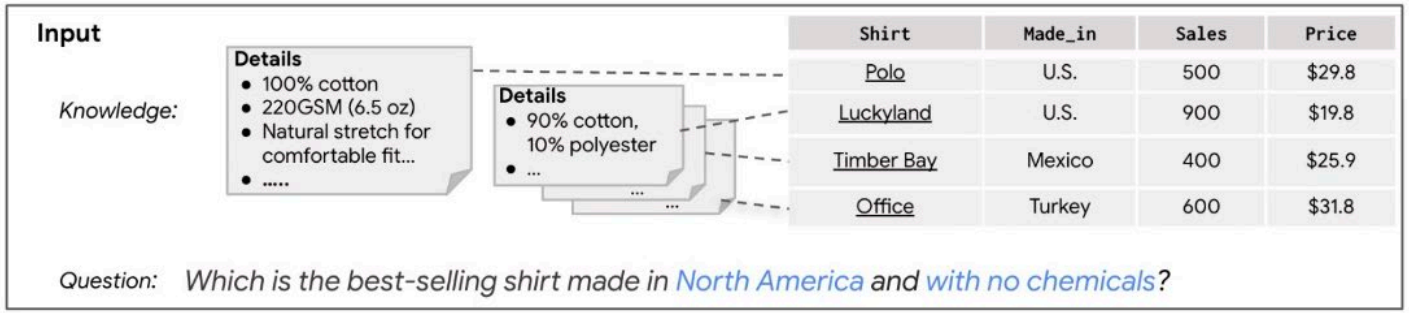
Zhoujun Cheng*[1,2], Tianbao Xie*[1], Peng Shi[5], Chengzu Li[1], Rahul Nadkarni[3], Yushi Hu[3], Caiming Xiong[6],
Dragomir Radev[7], Mari Ostendorf[3], Luke Zettlemoyer[3,8], Noah A. Smith[3,4], Tao Yu[1,3]
[1]The University of Hong Kong, [2]Shanghai Jiao Tong University, [3]University of Washington,
[4]Allen Institute for AI, [5]University of Waterloo, [6]Salesforce Research, [7]Yale University, [8]Meta AI

Paper Code Demo Twitter Video

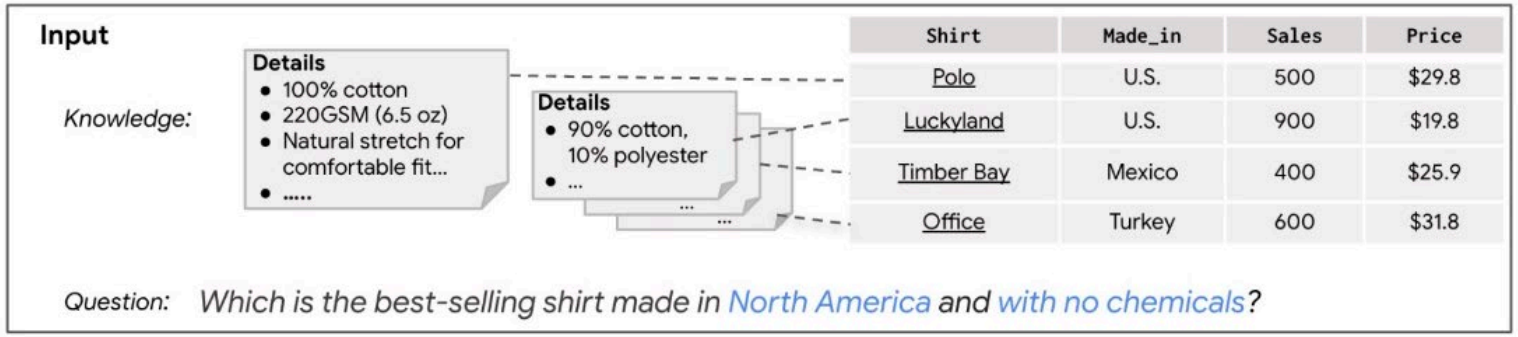
Project website: https://lm-code-binder.github.io, ICLR 2023

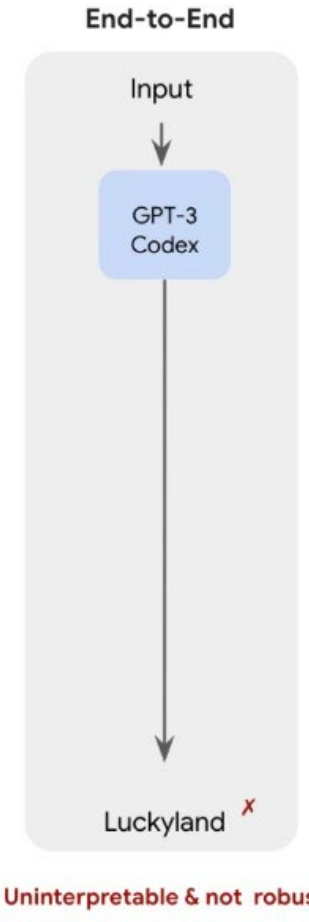
Example of LLMs + tool use in executable language grounding

BINDER is a training-free neural-symbolic framework that maps the task input to an executable BINDER program that
(1) allows binding API calls to GPT-3 Codex into SQL/Python,
(2) is executed with SQL/Python Interpreter + GPT-3 Codex to derive the answer.



LLM + no tool





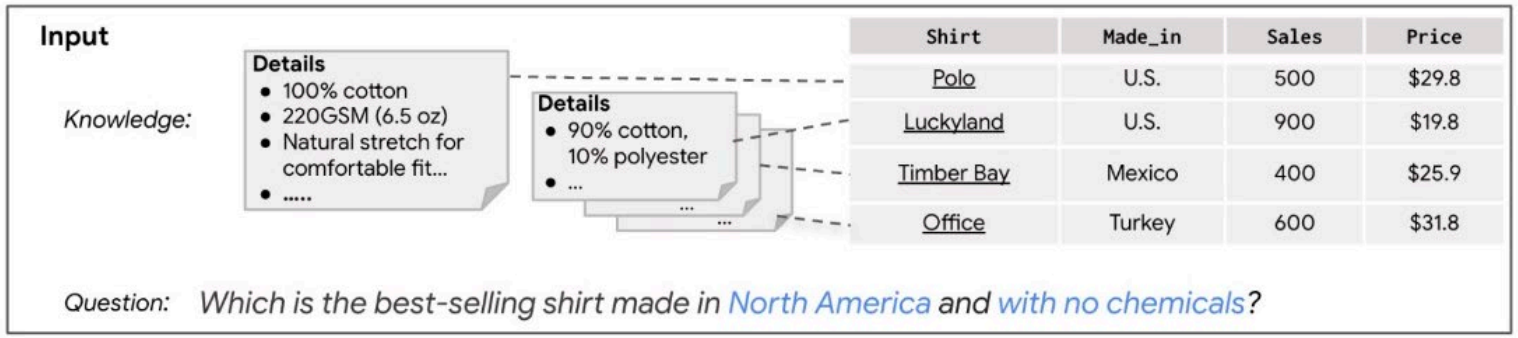
Pros

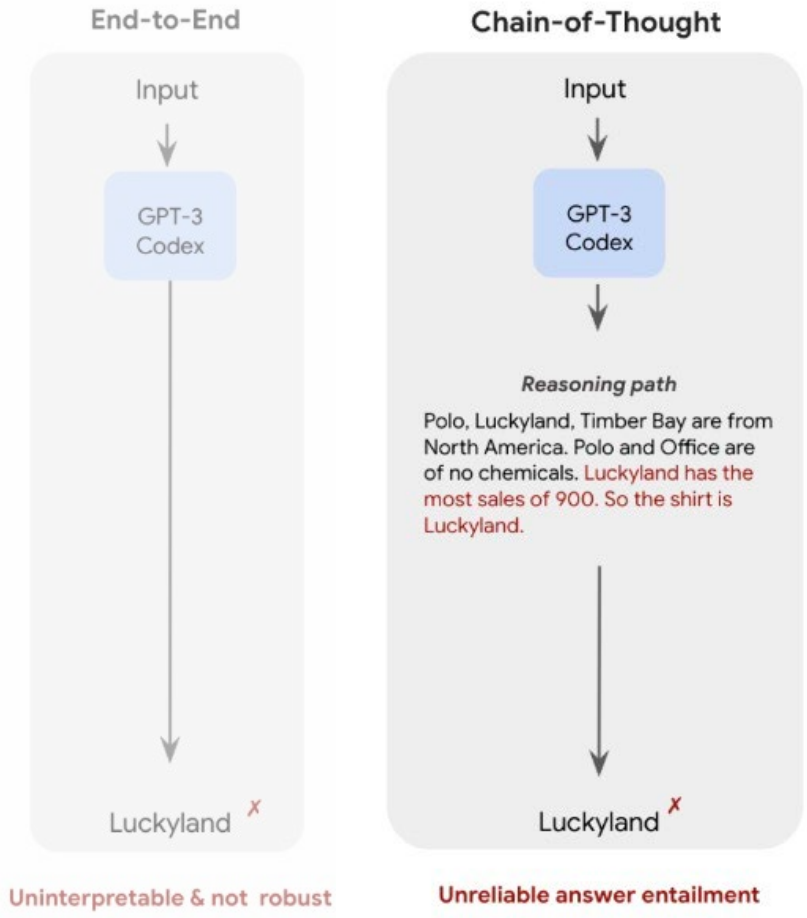
- General ✔

Cons

- Interpretable ✗
- Scalable ✗
- Robust ✗

LLM + no tool





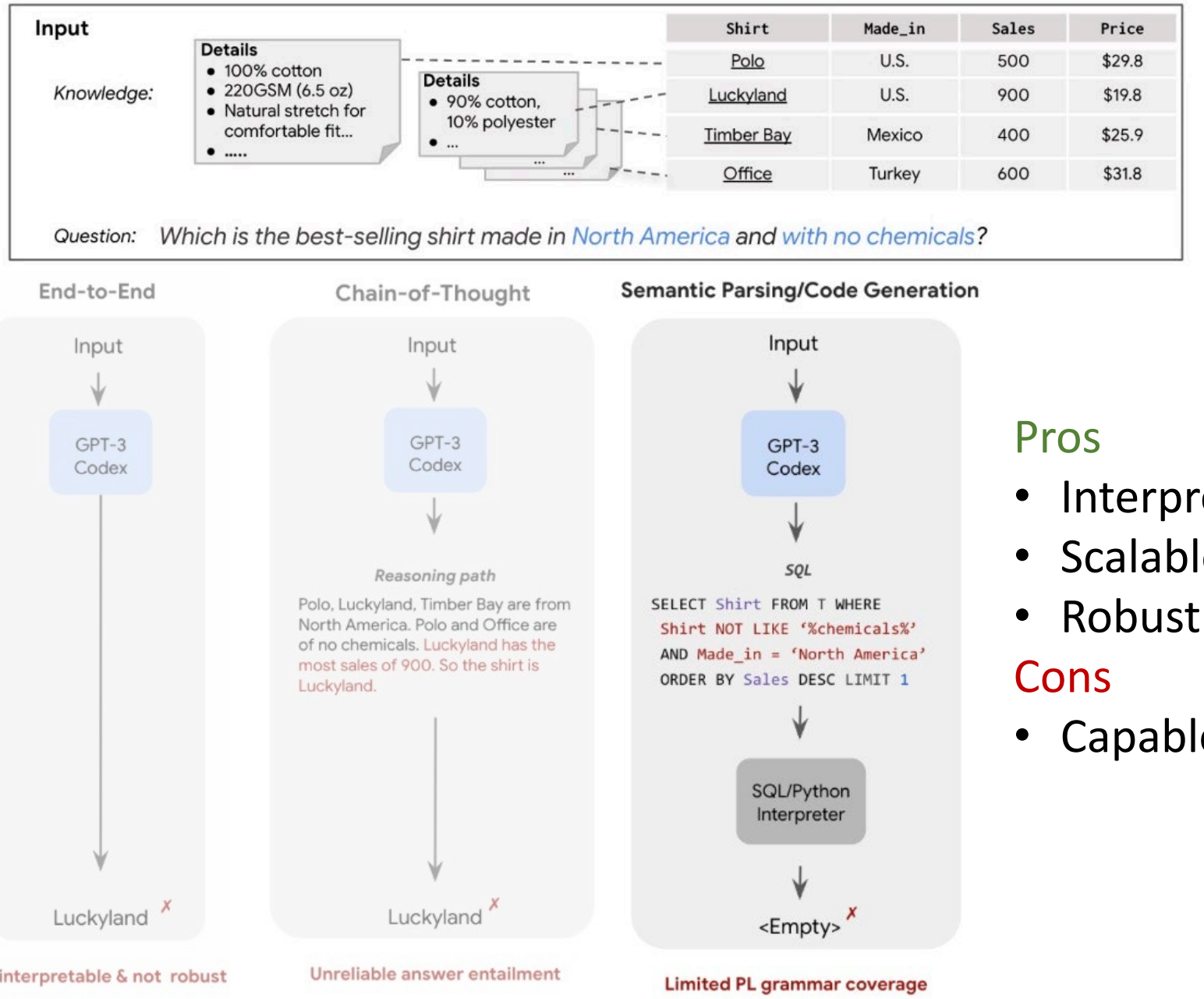
Pros

- Improved but still... ✔

Cons

- Interpretable: ✗
  - Unreliable answer entailment
- Scalable ✗
- Robust ✗

LLM + code



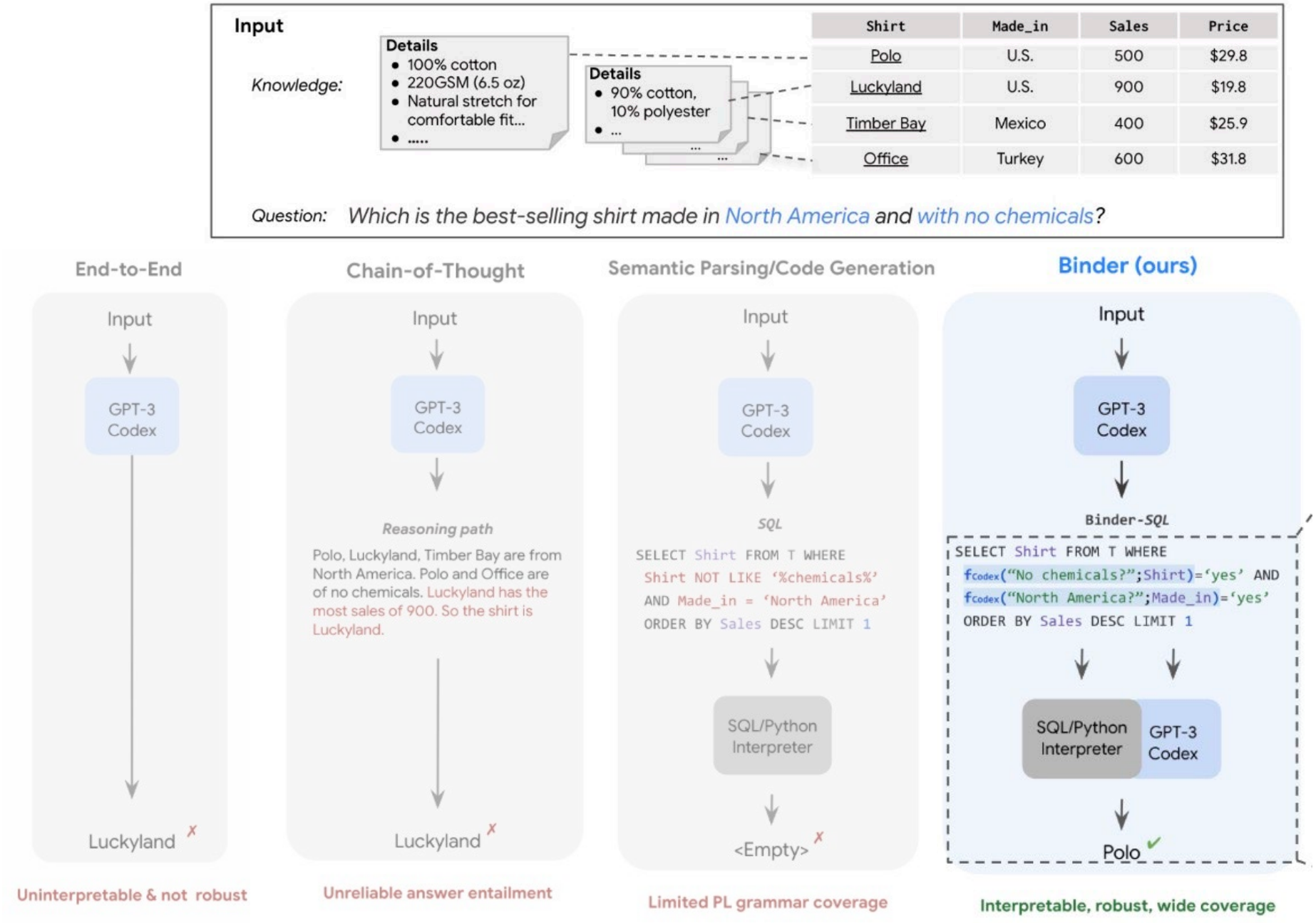
Pros

- Interpretable ✔
- Scalable ✔
- Robust ✔

Cons

- Capable ✘

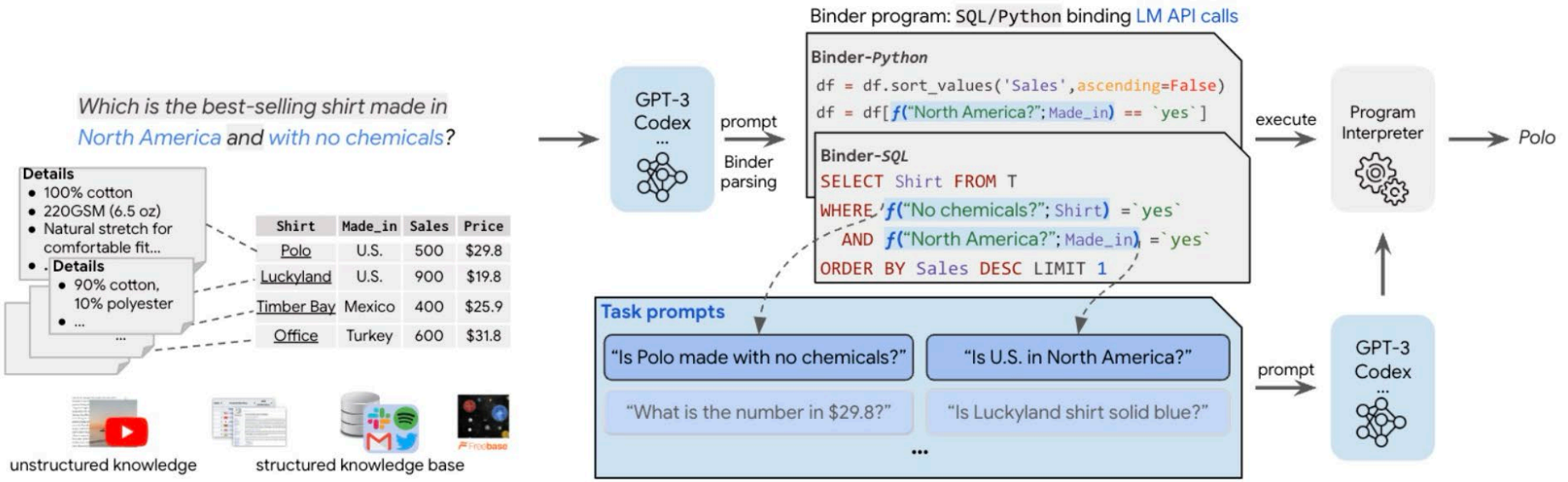
LLM + code and NLP expert function APIs



Pros

- Interpretable ✔
- Scalable ✔
- Robust ✔
- Capable ✔

LLM + code and NLP expert function APIs



BINDER first prompts Codex to parse a question input into a BINDER program, in which Codex has to decide (1) which parts in the input can be converted to the target programming language (grey clause in figure), (2) the corresponding task API calls (blue clause in figure) to prompt Codex to resolve the other parts, and (3) where to insert the API calls in the BINDER program.
Next, BINDER prompts Codex again to generate answers to the task API calls (given the generated task prompts), integrates the generated results back to the programming language, and executes the resulting programming language expression to derive the final answer.

LLM + code

ProgramAided Language models (PAL): uses the LLM to read natural language problems and generate programs as the intermediate reasoning steps, but offloads the solution step to a runtime such as a Python interpreter

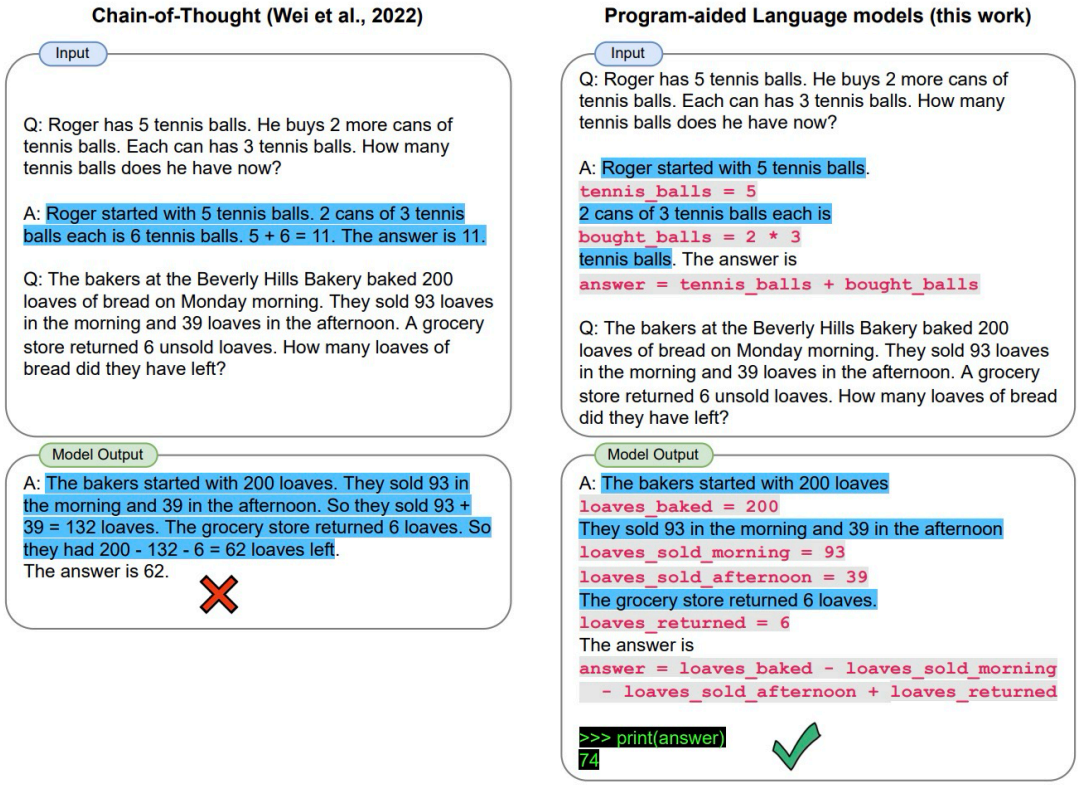


Figure 1: A diagram illustrating PAL: Given a mathematical reasoning question, Chain-of-thought (left) generates intermediate reasoning steps of free-form text. In contrast, Program-aided Language models (PAL, right) generate intermediate steps *and* Python code. This shifts the role of *running* the reasoning steps from the language model to the Python interpreter. The final answer is obtained by running the generated reasoning chain. Chain-of-thought reasoning is highlighted in blue; PAL steps are `highlighted in gray and pink`; the Python interpreter run is highlighted in black and green.

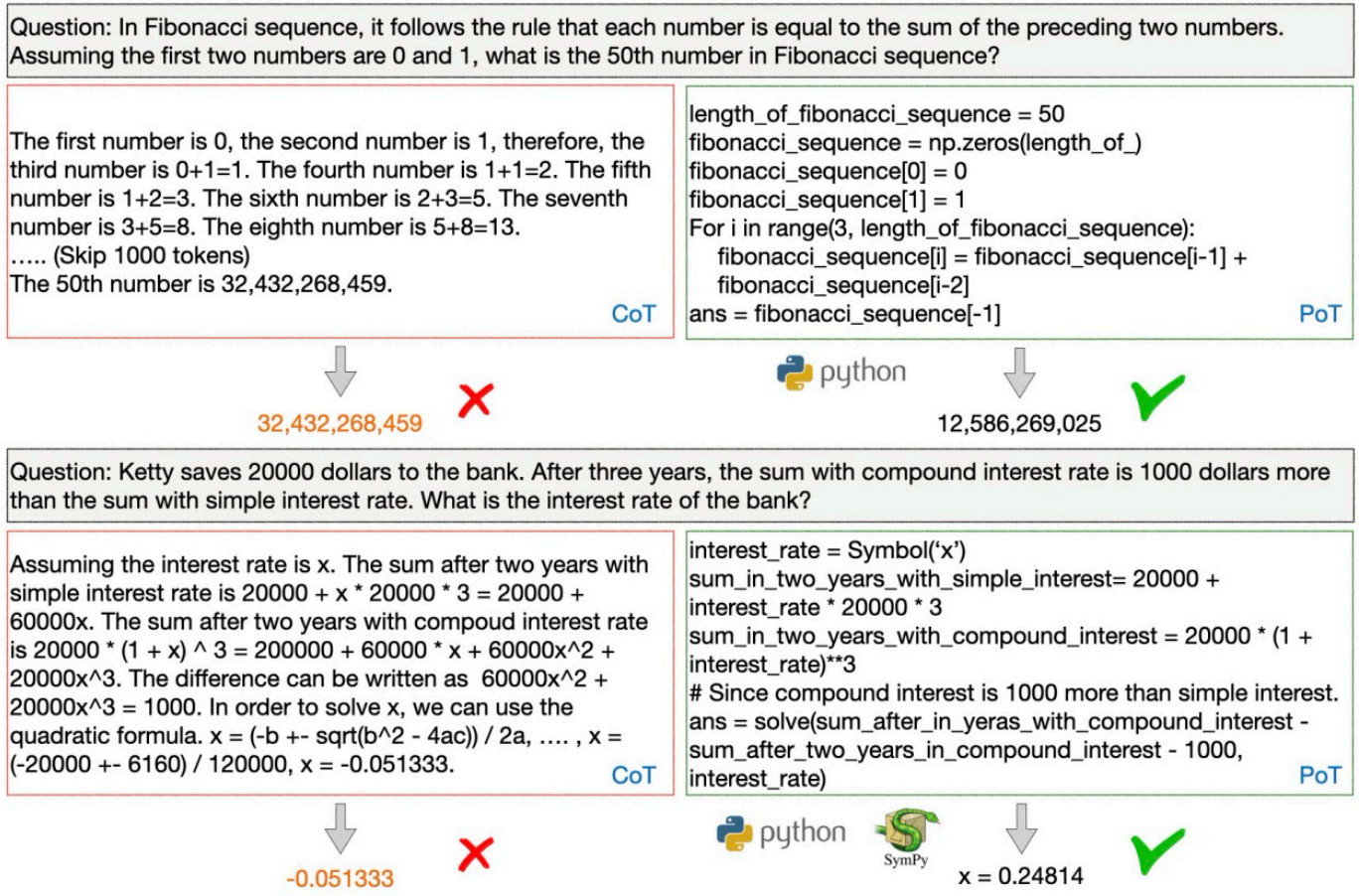


Figure 1: Comparison between Chain of Thoughts and Program of Thoughts.

[1] PAL: Program-aided Language Models

[2] Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks

LLM + APIs to search/browser for gathering information

WebGPT: Browser-assisted question-answering with human feedback

Reiichiro Nakano* Jacob Hilton* Suchir Balaji* Jeff Wu Long Ouyang

Christina Kim Christopher Hesse Shantanu Jain Vineet Kosaraju

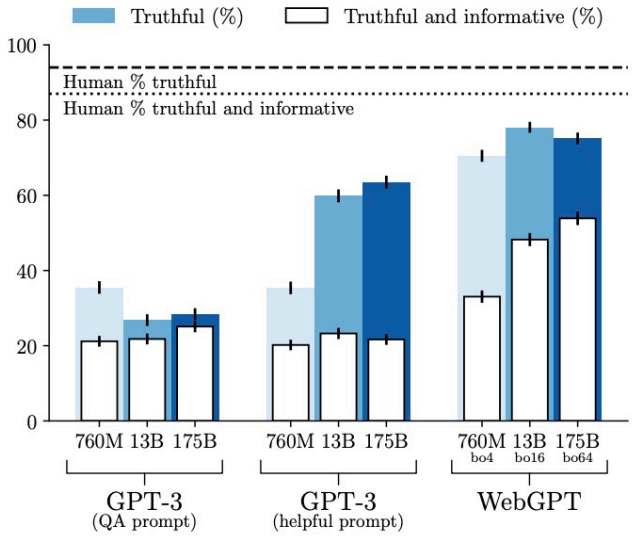
William Saunders Xu Jiang Karl Cobbe Tyna Eloundou Gretchen Krueger

Kevin Button Matthew Knight Benjamin Chess John Schulman

OpenAI

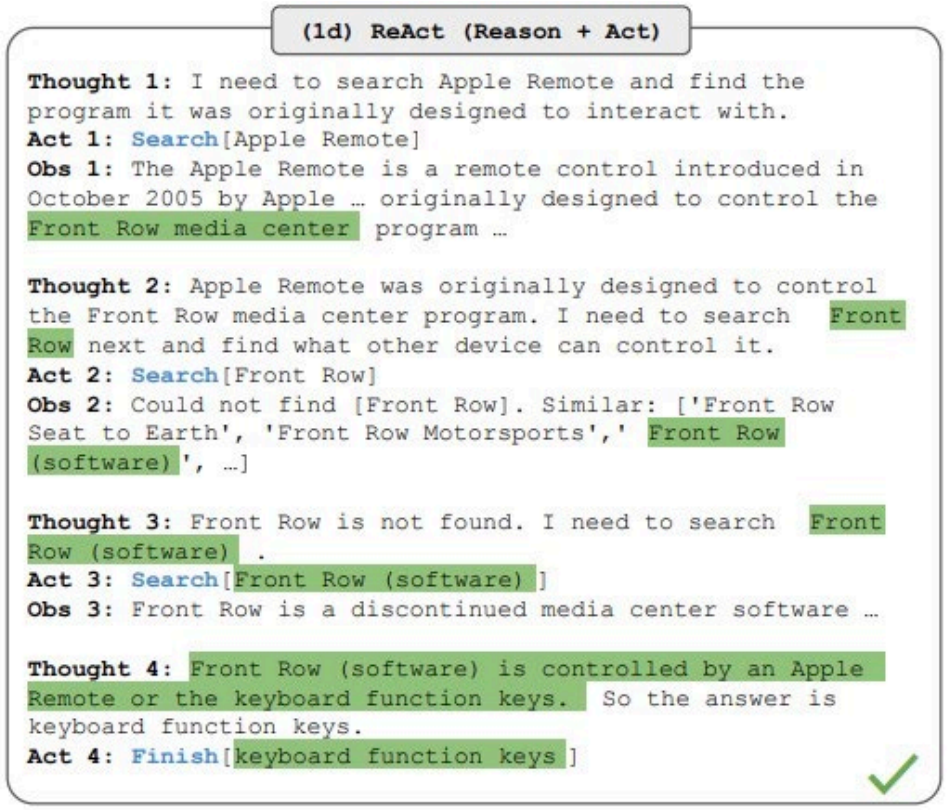
WebGPT fine-tunes GPT-3 to answer long-form questions using a text-based web browsing environment, which allows the model to search and navigate the web.

Command	Effect
`Search <query>`	Send `<query>` to the Bing API and display a search results page
`Clicked on link <link ID>`	Follow the link with the given ID to a new page
`Find in page: <text>`	Find the next occurrence of `<text>` and scroll to it
`Quote: <text>`	If `<text>` is found in the current page, add it as a reference
`Scrolled down <1, 2, 3>`	Scroll down a number of times
`Scrolled up <1, 2, 3>`	Scroll up a number of times
`Top`	Scroll to the top of the page
`Back`	Go to the previous page
`End: Answer`	End browsing and move to answering phase
`End: <Nonsense, Controversial>`	End browsing and skip answering phase

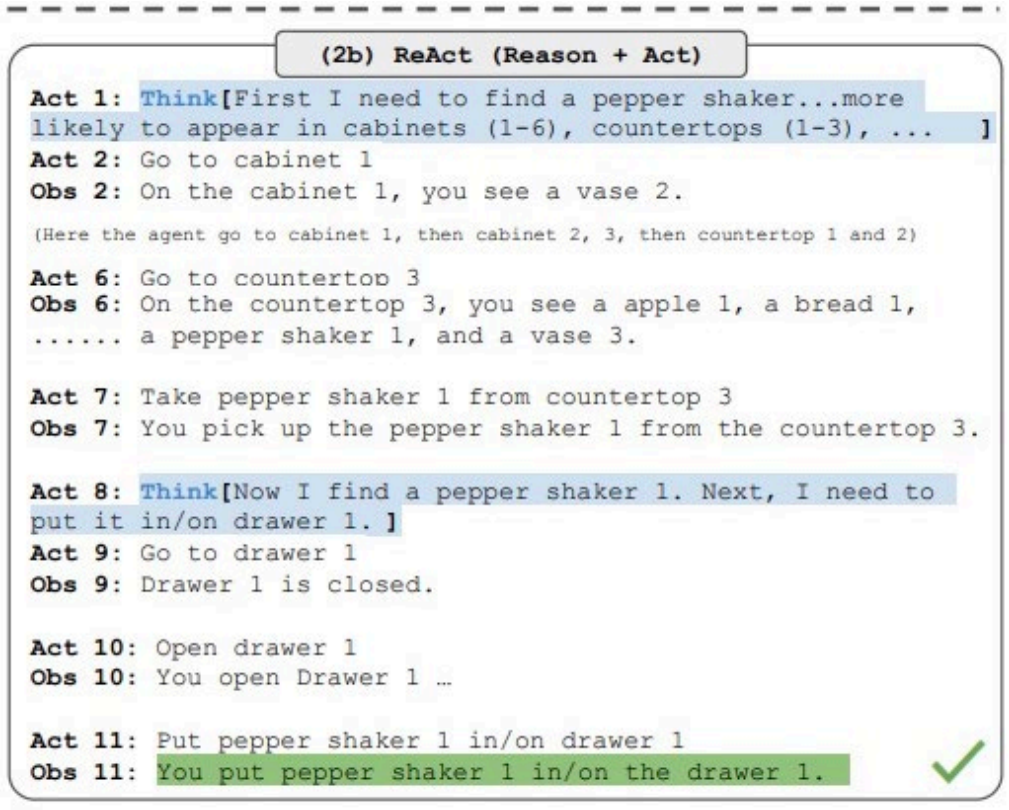


LLM + webs/apps or personalized functions

ReAct: generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments.



HotpotQA

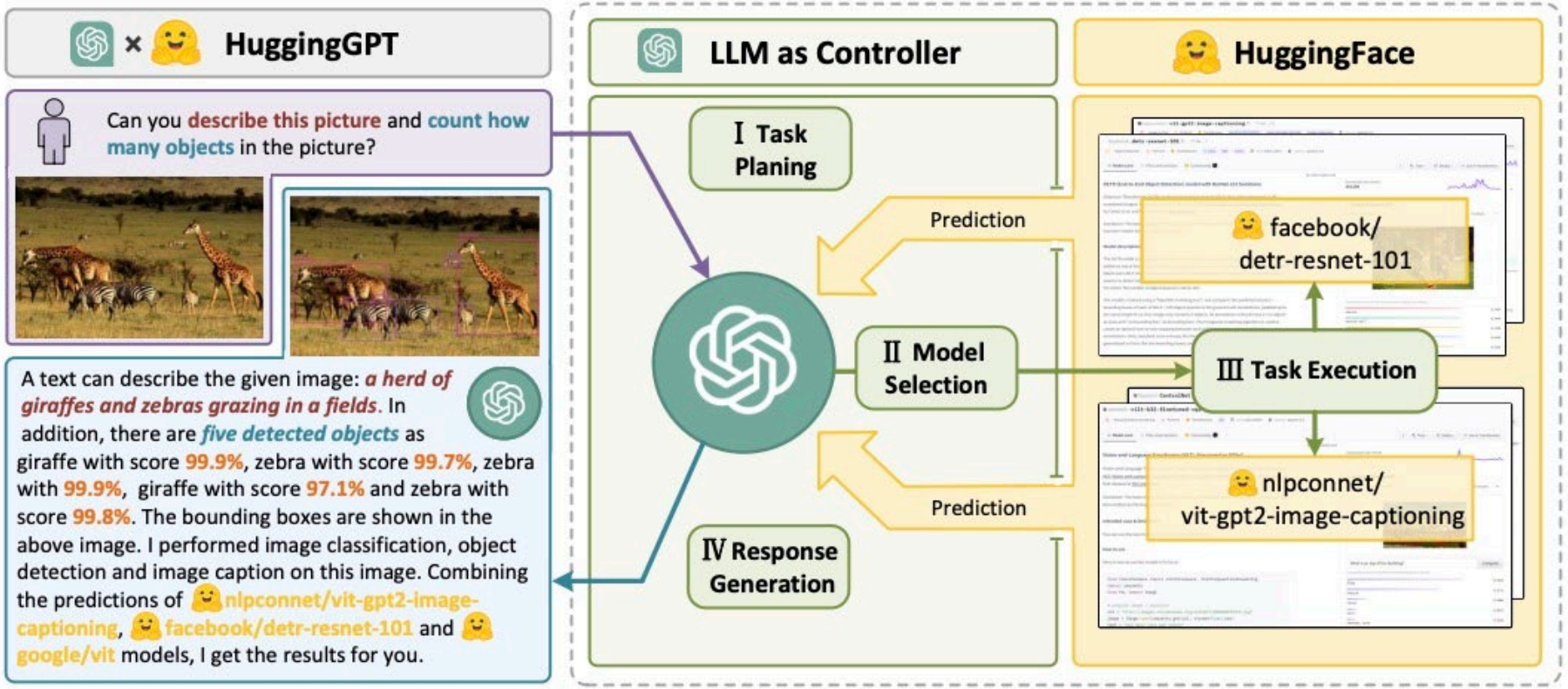


AlfWorld

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS, Google Brain

LLM + APIs to expert models

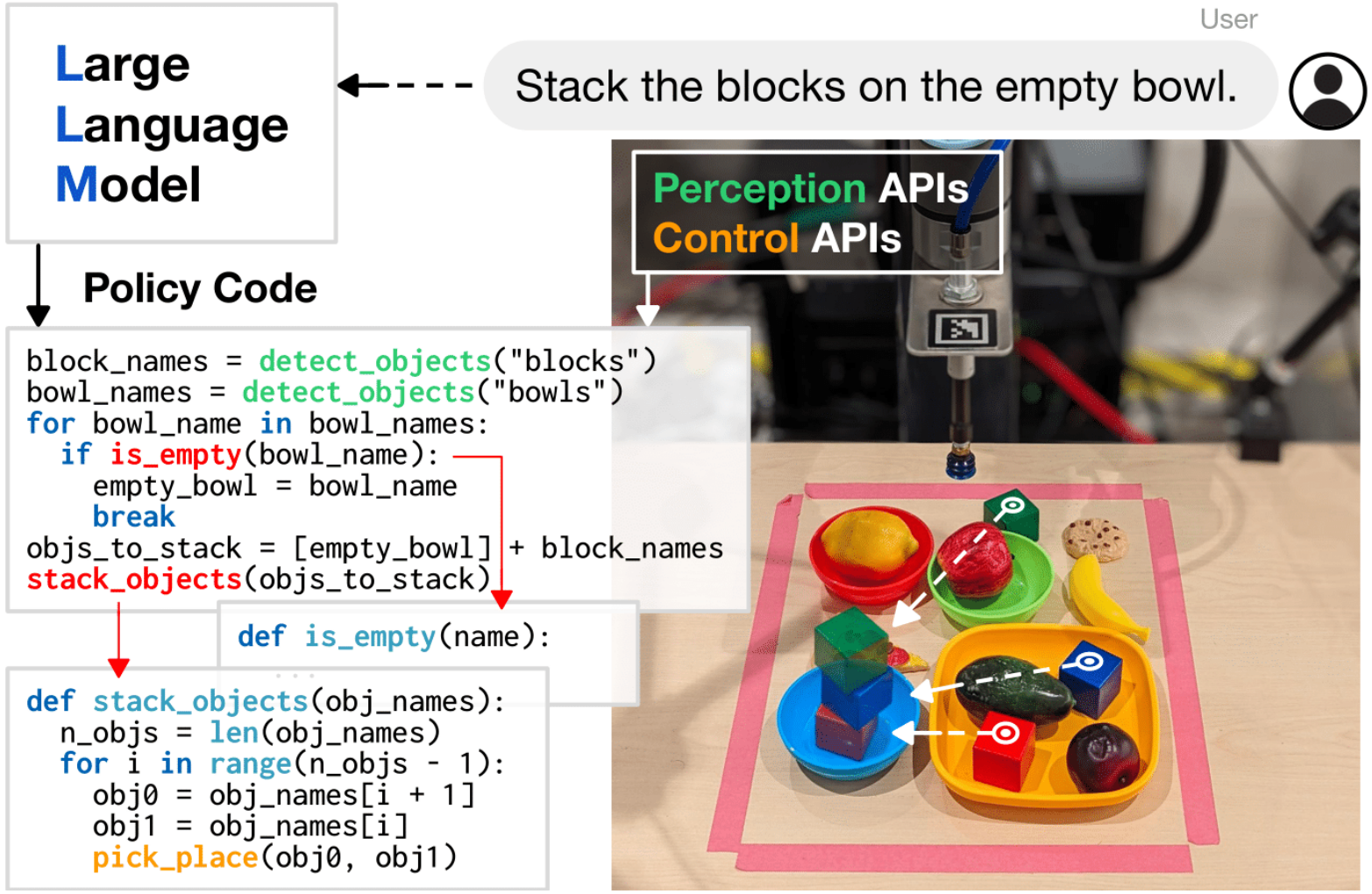
HuggingGPT, an LLM-powered agent that leverages LLMs (e.g., ChatGPT) to connect various AI models in machine learning communities (e.g., Hugging Face) to solve AI tasks. Specifically, we use ChatGPT to conduct task planning when receiving a user request, select models according to their function descriptions available in Hugging Face, execute each subtask with the selected AI model, and summarize the response according to the execution results



HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face, Microsoft

LLM + code, robotic arm, expert models: Code as Policies

Use LLM to write robot policy code given natural language commands.

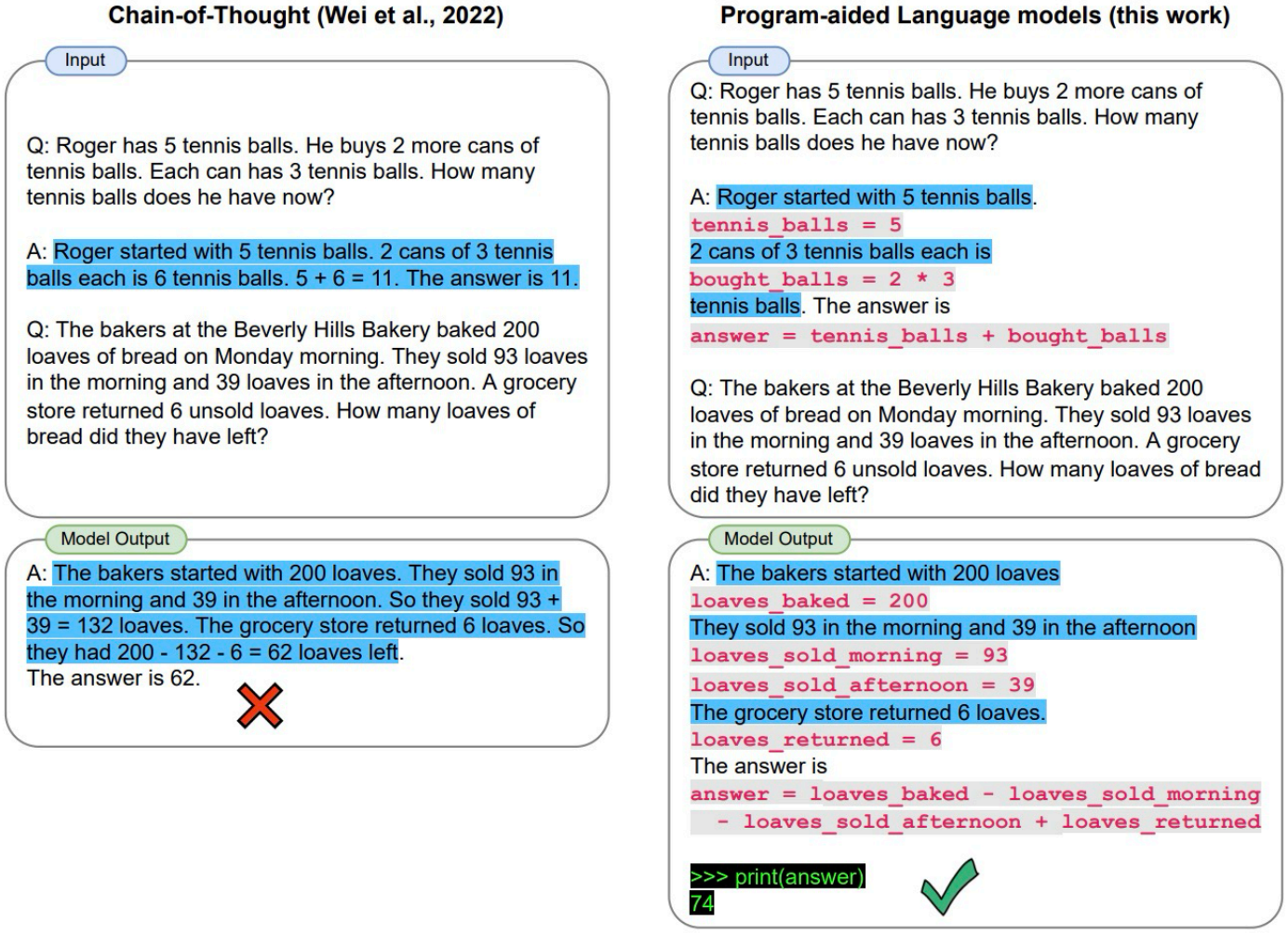


https://code-as-policies.github.io/

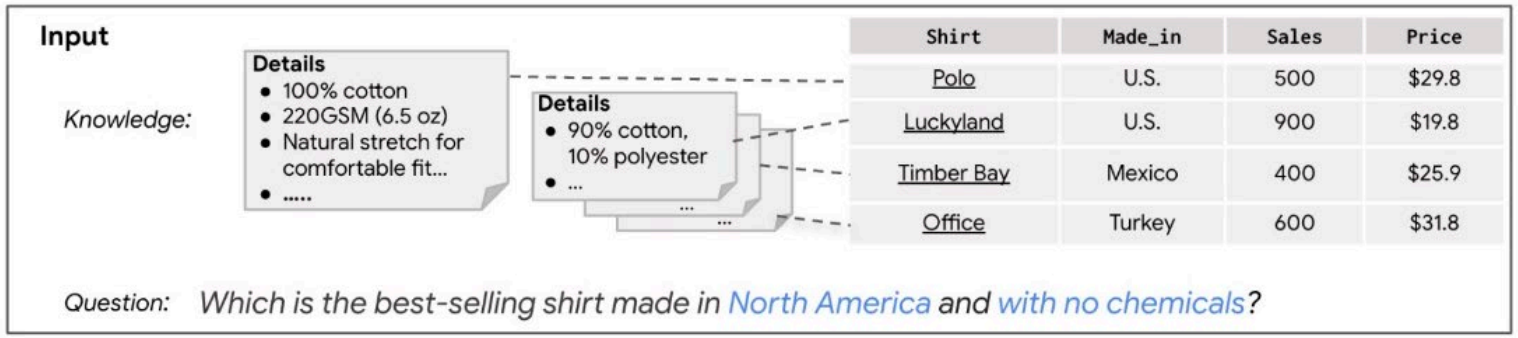
Methods for tool learning

LLM prompting for tool use

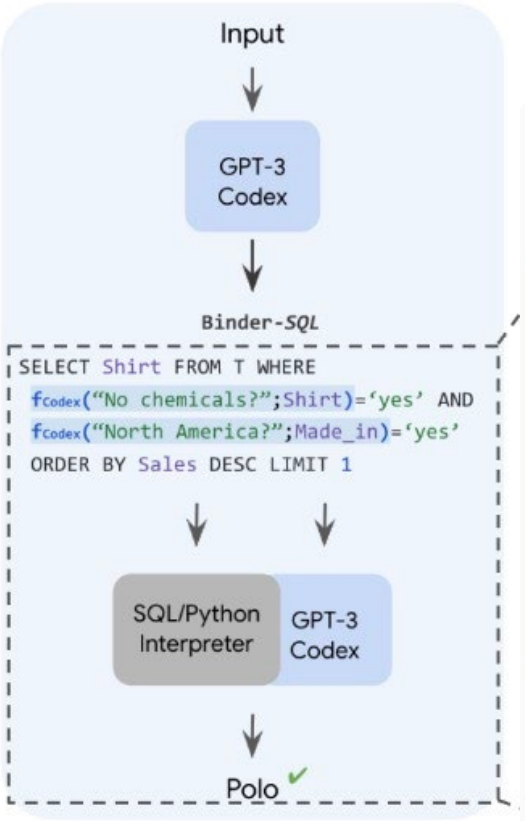
In a Program-aided Language model (PAL), the thoughts for a given natural language problem are generated as both interleaved natural language (NL) and programming language (PL) statements.



LLM prompting for tool use



Binder (ours)



Interpretable, robust, wide coverage

```
## WikiTQ
Generate SQL given the question and table to answer
the question correctly.
...

CREATE TABLE Electoral district of Lachlan(
    row_id int,
    member text,
    party text,
    term text)
/*
3 example rows:
SELECT * FROM w LIMIT 3;
row_id member party term
0 john ryan none 1859-1864
1 james martin none 1864-1869
2 james watson none 1869-1880
*/
Q: of the members of the third incarnation of the lachlan,
    who served the longest?
Binder: SELECT member FROM w ORDER BY
    f("How long does it last?"; term) DESC LIMIT 1
```

Task/tool description

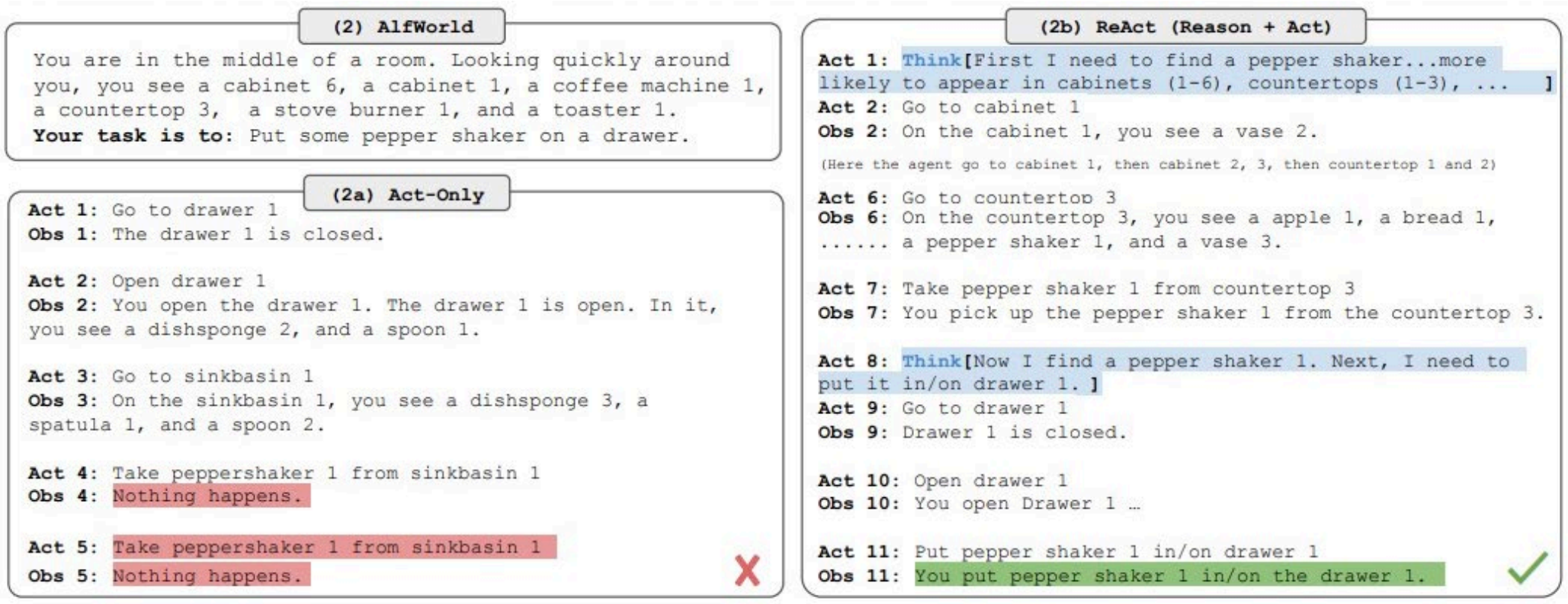
environment description

User question

Output code bounded with API calls

LLM + tool use for QA and decision making: ReAct

ReAct prompts LLMs to generate both verbal reasoning traces and actions pertaining to a task in an interleaved manner, which allows the model to perform dynamic reasoning to create, maintain, and adjust high-level plans for acting (reason to act), while also interact with the external environments (e.g. Wikipedia) to incorporate additional information into reasoning (act to reason).



LLM finetuning/pretraining for tool use: TALM

Tool Augmented Language Models (TALM), combining a text-only approach to augment language models with non-differentiable tools, and an iterative "self-play" technique to bootstrap performance starting from few tool demonstrations.

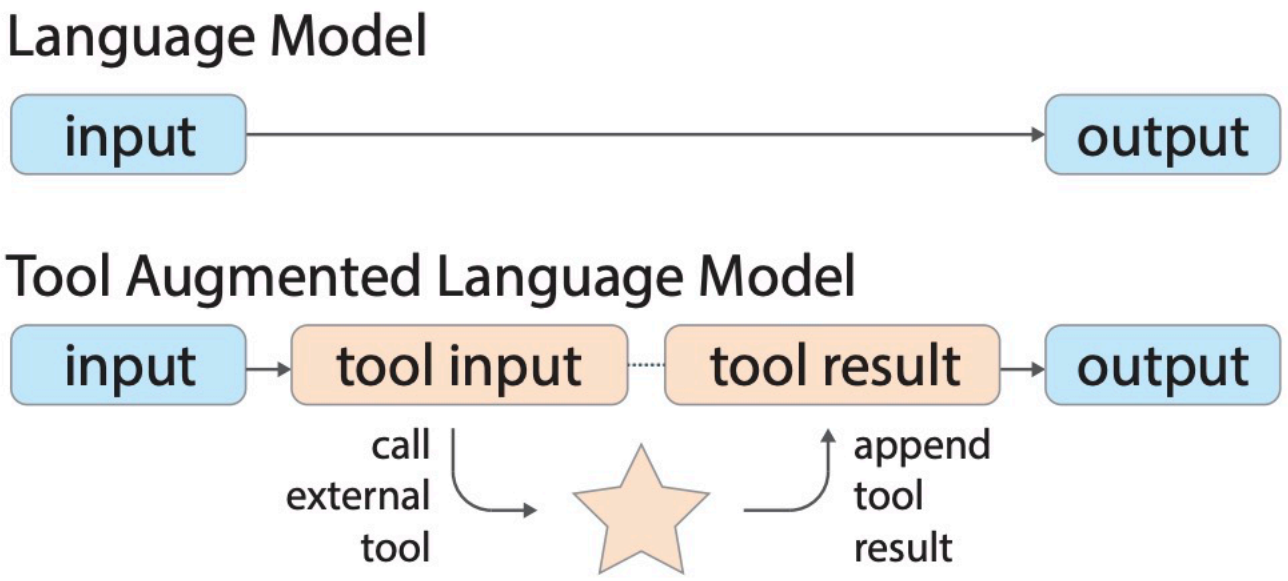


Figure 2: LM and Tool Augmented LMs.

Algorithm 1 Iterative Self-Play Algorithm.
x: task input, y: task output, t: tool input, r: tool output

$T = \{x_i, y_i\}_T$ # task set
$D = \{x_j, t_j, r_j, y_j\}_D$ # tool-use set
$P_\theta \leftarrow pretrained\ LM$
for $t \in [0, 1, ..., R]$ **do** # self-play rounds
 # finetune LM
  $\theta \leftarrow \underset{\theta}{\text{argmax}} \prod_D P_\theta(y_j|x_j, t_j, r_j) P_\theta(t_j|x_j)$
  for $x_i, y_i \in T$ **do** # iterate task set
    for $n \in [0, 1, ..., N]$ **do**
      $t_n \leftarrow P_\theta(t|x_i)$ # sample tool query
      $r_n \leftarrow Tool(t_n)$ # call tool API
      $y_n \leftarrow P_\theta(y|x_i, t_n, r_n)$ # get task output
      if $|y_n - y_i| < th$ **then** # filter wrong output
        $D \leftarrow D \cup \{x_i, t_n, r_n, y_n\}_1$
 # update tool-use set

LLM finetuning/pretraining for tool use: Toolformer

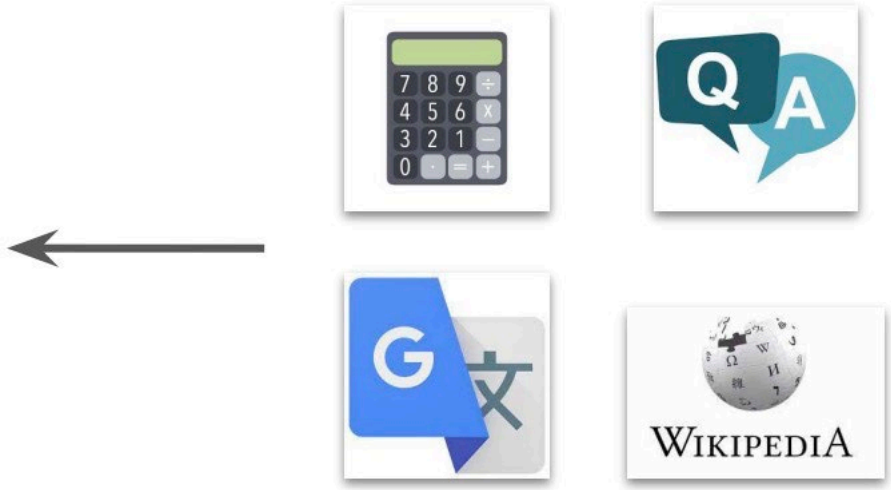
Toolformer is a model trained to decide which APIs to call, when to call them, what arguments to pass, and how to best incorporate the results into future token prediction. This is done in a self-supervised way, requiring nothing more than a handful of demonstrations for each API.

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: Language Models Can Teach Themselves to Use Tools, Meta AI

LLM finetuning/pretraining for tool use: Toolformer

The use of tools should be learned in a self-supervised way without requiring large amounts of human annotations. The dataset construction pipeline:

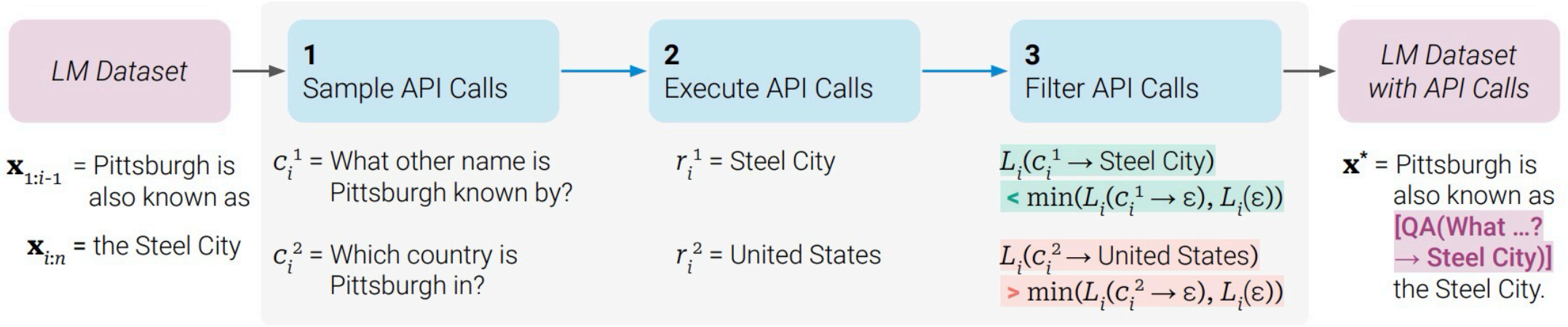


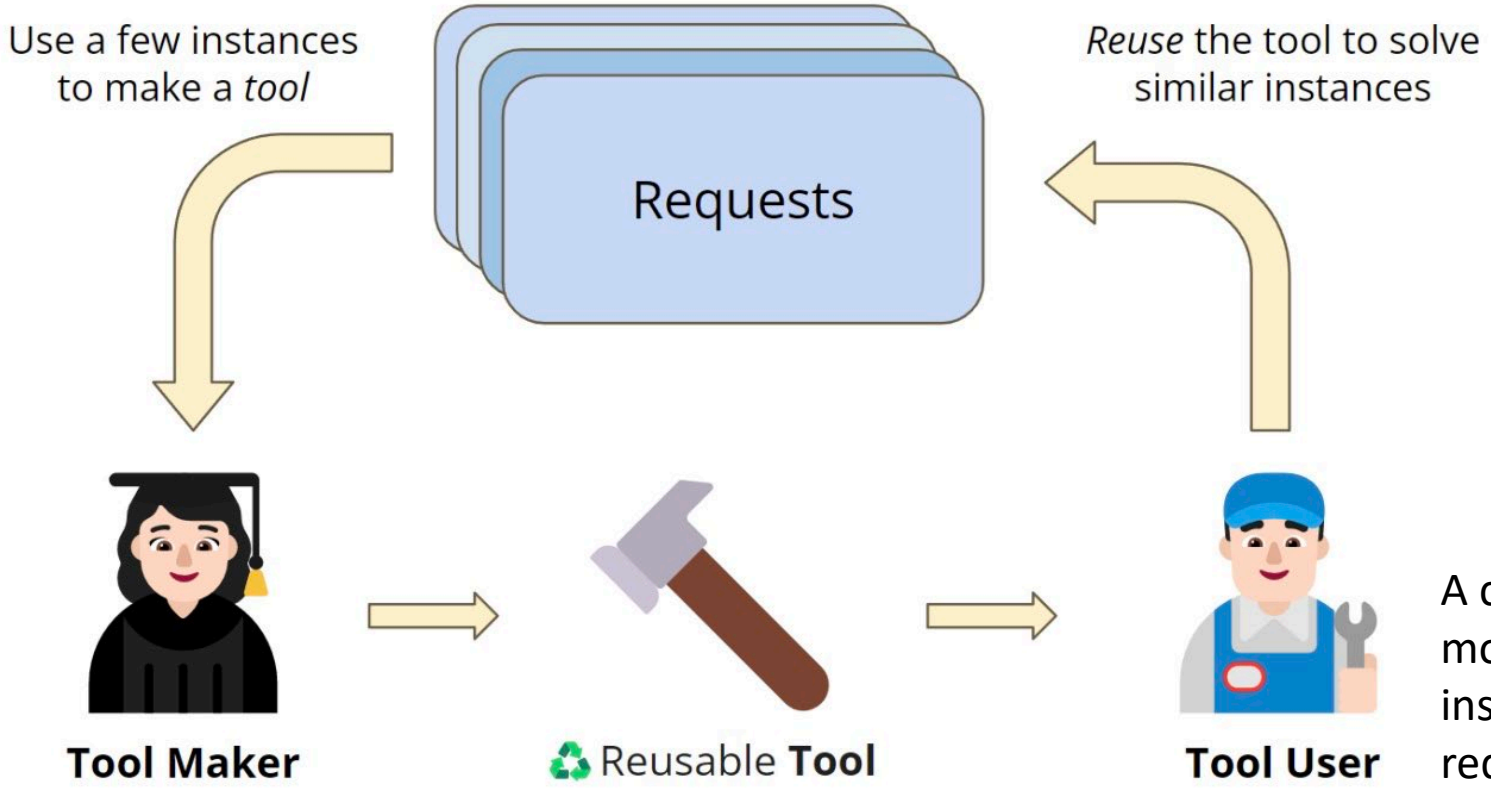
Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text $\mathbf{x}$, we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \ldots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text $\mathbf{x}^*$.

Toolformer: Language Models Can Teach Themselves to Use Tools, Meta AI

Other recent works

Other recent related work: LLM as a tool maker

In situations with numerous problem-solving requests, directly utilizing a powerful LLM to solve all the instances can result in high costs. On the other hand, lightweight models are cost-effective but usually struggle with complex tasks.



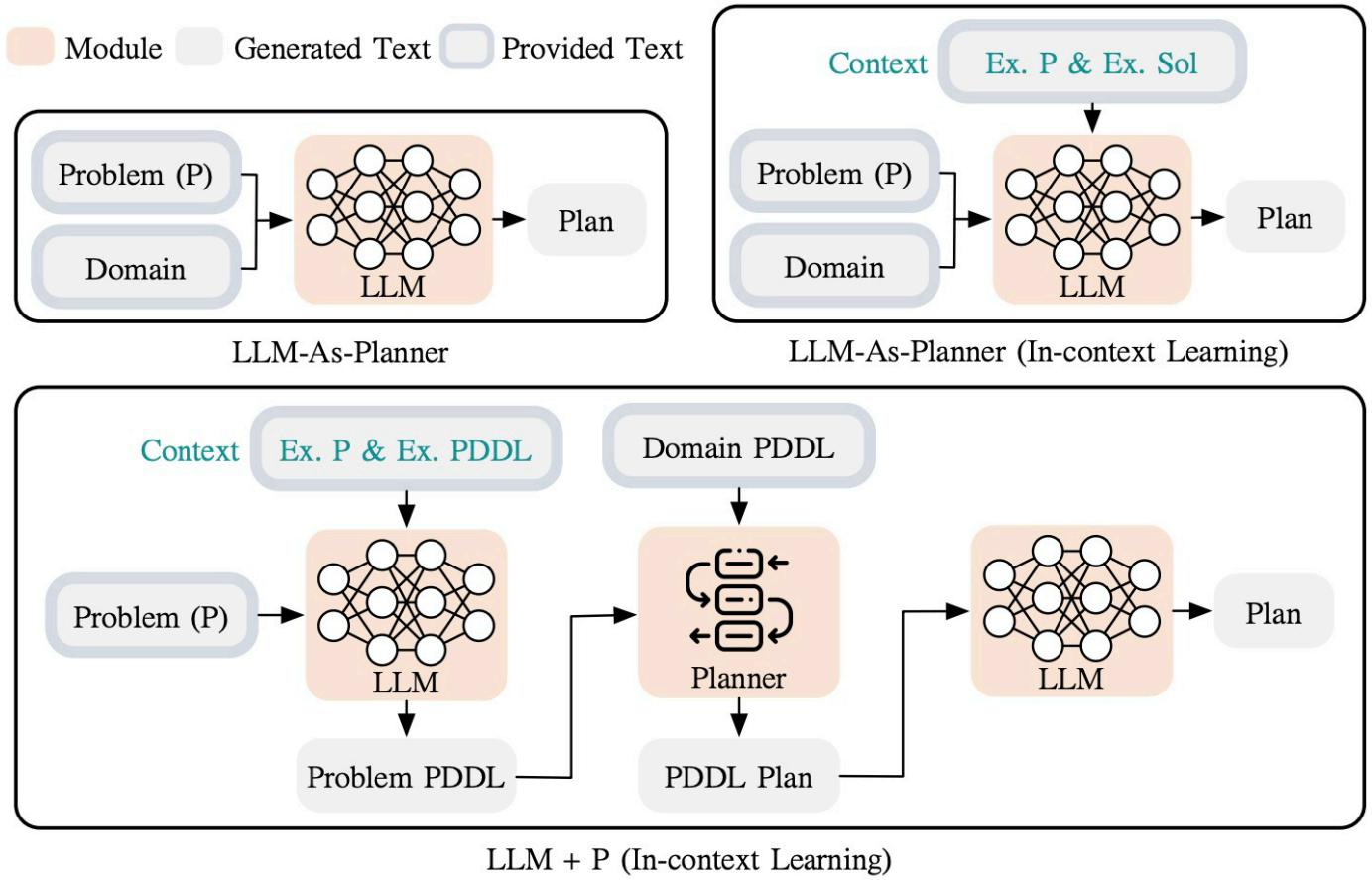
A powerful model as the tool maker to generate reusable tools (implemented as Python functions)

A cost-effective tool user model for solving similar instances in the following requests

LARGE LANGUAGE MODELS AS TOOL MAKERS, Google Deepmind

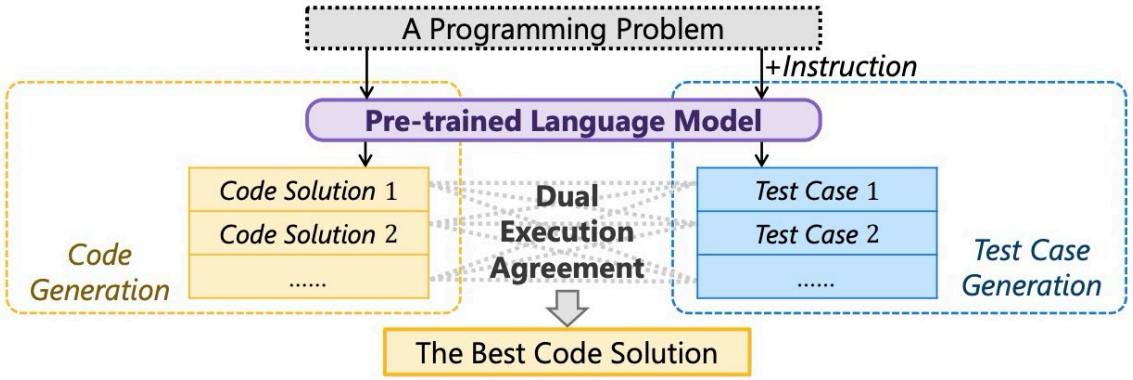
Other recent related work: LLM as a tool maker

LLMs cannot reliably solve long-horizon robot planning problems. By contrast, classical planners, once a problem is given in a formatted way, can use efficient search algorithms to quickly identify correct, or even optimal, plans. LLM+P first converts the language description into a file written in the planning domain definition language (PDDL), then leveraging classical planners to quickly find a solution, and then translating the found solution back into natural language

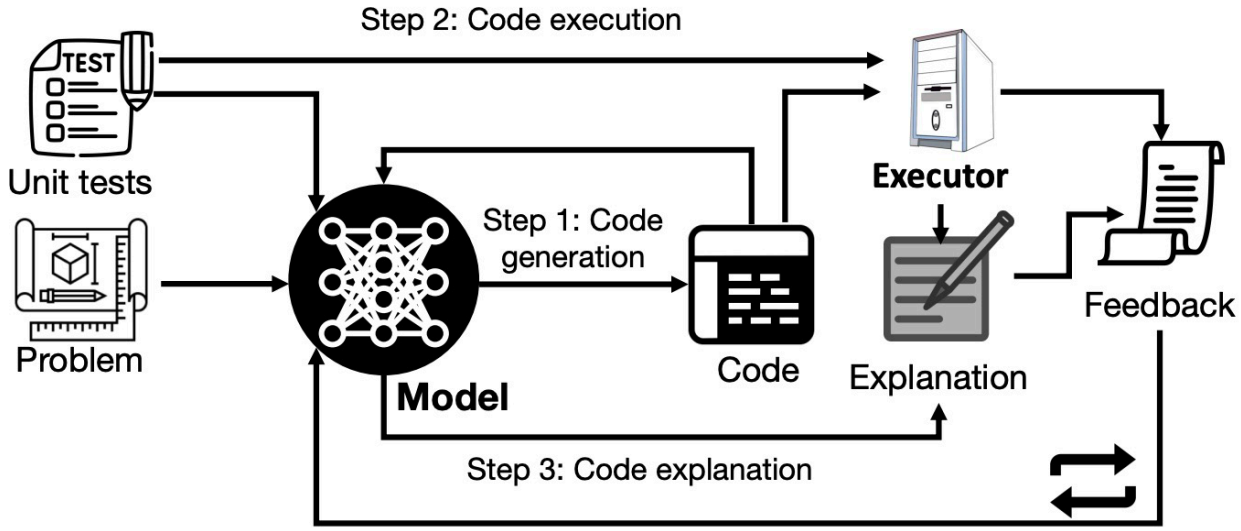


LLM+P: Empowering Large Language Models with Optimal Planning Proficiency

Other recent related work: code generation



CODET: both the code solutions and the test cases are generated by the pre-trained language model. The best code solution is then selected by a dual execution agreement.

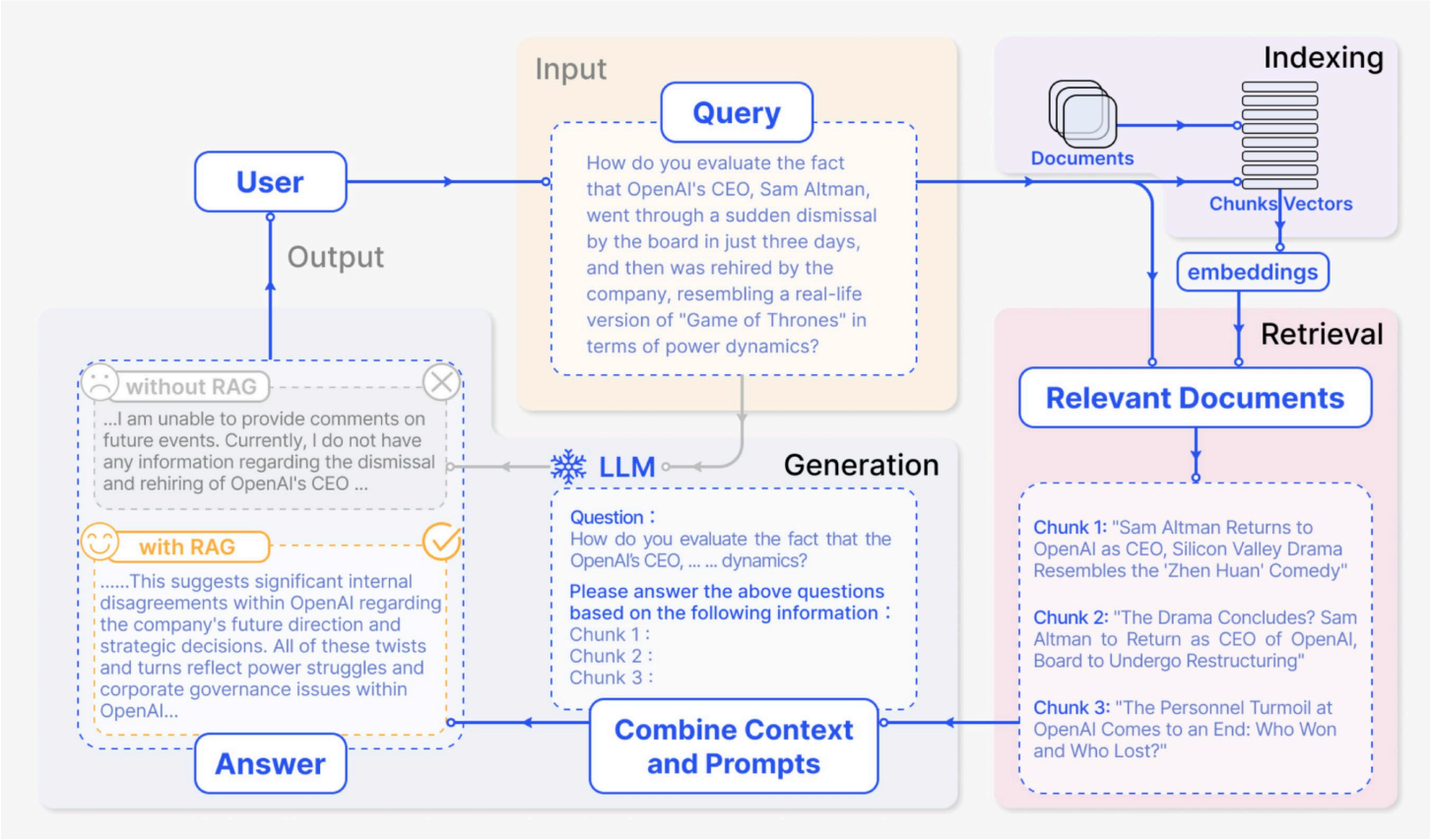


SELF-DEBUGGING: teaches a large language model to debug its predicted program via few-shot demonstrations. The model is able to identify its mistakes by investigating the execution results and explaining the generated code in natural language.

[1] CODET: CODE GENERATION WITH GENERATED TESTS, Microsoft
[2] TEACHING LARGE LANGUAGE MODELS TO SELFDEBUG, Google DeepMind

Incorporating external knowledge-RAG

- Vector Database
  - Embedding
  - Indexing
  - Querying (Retrieve)
  - Post-process
- Generator
  - LLM (like GPT, DeepSeek)



Challenges and future work

- Complexity: more complex domain professional/unseen tools?
- Interactivity: go beyond single turn?
- Evaluation: multiple possible solutions? Real-time interactive evaluation?
- Efficiency: smaller models?
- Reliability: know when to abstain, know its capacity, memorizing and querying tools?
- Others
  - Better tool API design/tool making?
  - Personalization?
  - ……

References

[1] Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents

[2] Do As I Can, Not As I Say: Grounding Language in Robotic Affordances

[3] Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language

[4] TALM: Tool Augmented Language Models

[5] Inner Monologue: Embodied Reasoning through Planning with Language Models

[6] JARVIS: A Neuro-Symbolic Commonsense Reasoning Framework for Conversational Embodied Agents

[7] ProgPrompt: Generating Situated Robot Task Plans using Large Language Models

[8] Code as Policies: Language Model Programs for Embodied Control

[9] Binding Language Models in Symbolic Languages

[10] Synergizing Reasoning and Acting in Language Models

[11] Code4Struct: Code Generation for Few-Shot Event Structure Prediction

[12] Mind's Eye: Grounded Language Model Reasoning through Simulation

[13] PAL: Program-aided Language Models

[14] Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks

[15] Planning with Large Language Models via Corrective Re-prompting

References

[16] Augmented Language Models: a Survey

[17] LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models

[18] Don't Generate, Discriminate: A Proposal for Grounding Language Models to Real-World Environments

[19] Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning

[20] Toolformer: Language Models Can Teach Themselves to Use Tools

[21] Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents

[22] Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning

[23] Grounded Decoding: Guiding Text Generation with Grounded Models for Robot Control

[24] PaLM-E: An Embodied Multimodal Language Model

[25] ViperGPT: Visual Inference via Python Execution for Reasoning

[26] Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models

[27] HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace

[28] TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs

[29] ART: Automatic multi-step reasoning and tool-use for large language models

[30] API-Bank: A Benchmark for Tool-Augmented LLMs

References

[31] Tool Learning with Foundation Models

[32] Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models

[33] GeneGPT: Augmenting Large Language Models with Domain Tools for Improved Access to Biomedical Information

[34] LLM as A Robotic Brain: Unifying Egocentric Memory and Control

[35] Voyager: An Open-Ended Embodied Agent with Large Language Models

[36] GPT4Tools: Teaching Large Language Model to Use Tools via Self-instruction

[37] PEARL: Prompting Large Language Models to Plan and Execute Actions Over Long Documents

[38] Large Language Models as Tool Makers

[39] Gorilla: Large Language Model Connected with Massive APIs

[40] On the Tool Manipulation Capability of Open-source Large Language Models

[41] Making Language Models Better Tool Learners with Execution Feedback

[42] Small models are valuable plug-ins for large language models

[43] Hierarchical Prompting Assists Large Language Model on Web Navigation

[44] Multimodal Web Navigation with Instruction-Finetuned Foundation Models

[45] ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings

[46] CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing

[47] SheetCopilot: Bringing Software Productivity to the Next Level through Large Language Models

References

[48] SPRING: GPT-4 Out-performs RL Algorithms by Studying Papers and Reasoning

[49] SQL-PaLM: Improved Large Language ModelAdaptation for Text-to-SQL

[50] From Pixels to UI Actions: Learning to Follow Instructions via Graphical User Interfaces

[51] Modular Visual Question Answering via Code Generation

[52] ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases

[53] Mind2Web: Towards a Generalist Agent for the Web

[54] LLM+P: Empowering Large Language Models with Optimal Planning Proficiency

[55] Search-in-the-Chain: Towards Accurate, Credible and Traceable Large Language Models for Knowledge-intensive Tasks

[56] OpenAGI: When LLM Meets Domain Experts

[57] Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs

[58] Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning

[59] ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models

[60] ToolCoder: Teach Code Generation Models to use API search tools

[61] Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models

[62] Can Language Models Teach Weaker Agents? Teacher Explanations Improve Students via Theory of Mind