

Neuro-Symbolic Approaches: Large Language Models + Tool Use

Tutorial on Complex Reasoning over Natural Language



Tao Yu
University of Hong Kong
07/09/2023

Language models are powerful, but they still suffer from

- Lack of interpretability
- Inconsistency
- Limited scalability
- Restricted capabilities
- ...

About this tutorial



: Can LLMs utilize external tools to not only expand their capacities but also to make our NLP systems more robust, scalable, and interpretable?



: Neuro-symbolic approaches in the era of large language models (LLMs): LLMs + tool use or tool augmented LLMs.

Augmented Language Models: a Survey

<p>Grégoire Mialon* Roberto Dessì† Maria Lomeli* Christoforos Nalmpantis* Ram Pasunuru* Roberta Raleanu* Baptiste Rozière* Timo Schick* Jane Dwivedi-Yu* Asli Celikyilmaz* Edouard Grave* Yann LeCun* Thomas Scialom*</p>	<p>gmalon@meta.com rdessi@meta.com marialomeli@meta.com christoforos@meta.com rpasunaru@meta.com raleanu@meta.com broz@meta.com schick@meta.com janeyu@meta.com aslic@meta.com egrave@meta.com yann@meta.com tscialom@meta.com</p>
---	--

*Meta AI †Universitat Pompeu Fabra

Tutorial agenda

- Motivations
- Introduction to LLM + tool use
 - LLM + tool use in the perspective of executable language grounding
 - LLM + tool use examples
- Recent methods of LLM + tool use
 - LLM prompting for tool use
 - LLM finetuning/pretraining for tool use
- Other recent related work
 - Tool making
 - Planning
 - Code generation
- Challenges and future work

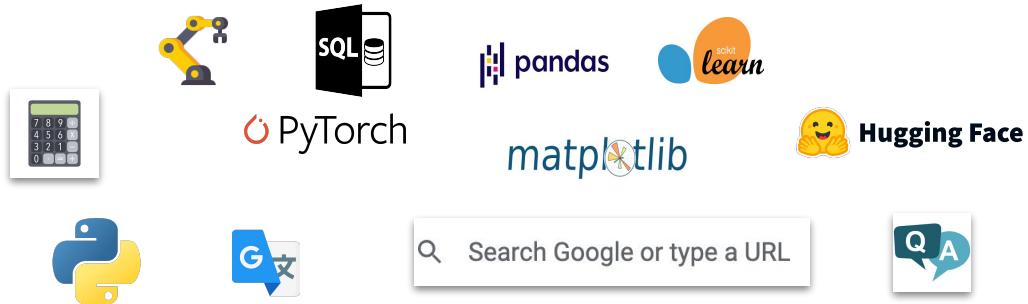
Human + tool use: motivations

- 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

- 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 and productivity

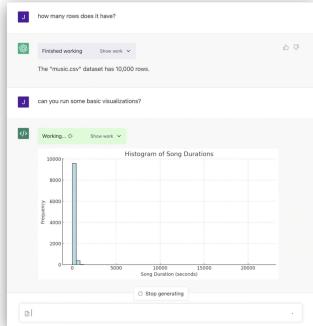


Search Google or type a URL

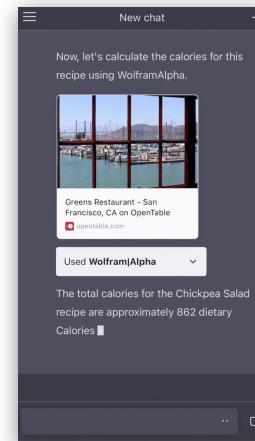
LLMs + tool use in perspective of executable language grounding

Ground language models into **executable actions**

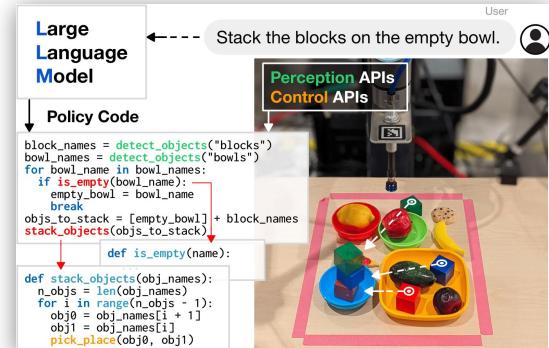
- Mapping natural language instructions into code or actions executable within various environments such as databases, web applications, and robotic physical world.
- LM (planning and reasoning) + **actions**



Data analysis



Web/Apps



Robotic physical world

LLMs + tool use in perspective of executable language grounding

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 seq 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

Binder: Binding Language Models in Symbolic Languages

Zhoujun Cheng^{*1,2}, Tianbao Xie^{*1}, Peng Shi⁵, Chengzu Li¹, Rahul Nadkarni³, Yushi Hu³, Caiming Xiong⁶,
Dragomir Radev⁷, Mari Ostendorf³, Luke Zettlemoyer^{3,8}, Noah A. Smith^{3,4}, Tao Yu^{1,3}

¹The University of Hong Kong, ²Shanghai Jiao Tong University, ³University of Washington,

⁴Allen Institute for AI, ⁵University of Waterloo, ⁶Salesforce Research, ⁷Yale University, ⁸Meta AI



arXiv



Code



Demo



Twitter



Video

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

Input

Knowledge:

Details

- 100% cotton
- 220GSM (6.5 oz)
- Natural stretch for comfortable fit...
-

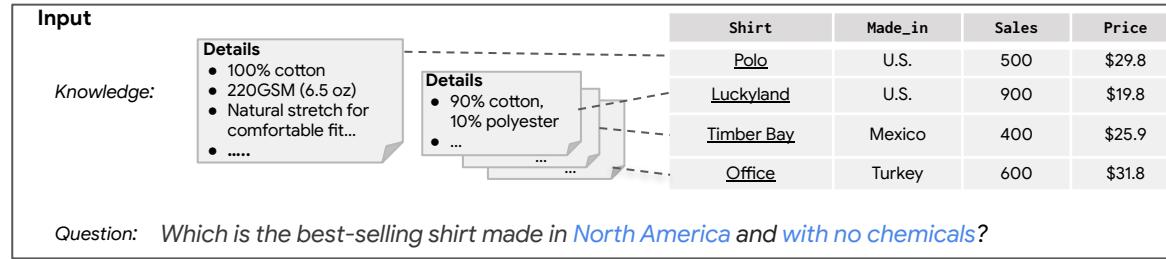
Details

- 90% cotton, 10% polyester
- ...

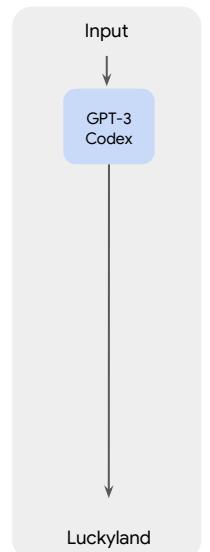
Shirt	Made_in	Sales	Price
<u>Polo</u>	U.S.	500	\$29.8
<u>Luckyland</u>	U.S.	900	\$19.8
<u>Timber Bay</u>	Mexico	400	\$25.9
<u>Office</u>	Turkey	600	\$31.8

Question: Which is the best-selling shirt made in *North America* and *with no chemicals*?

LLM + no tool



End-to-End



Pros

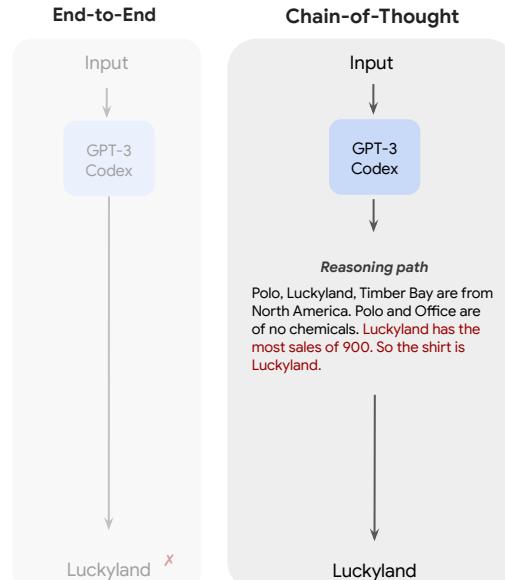
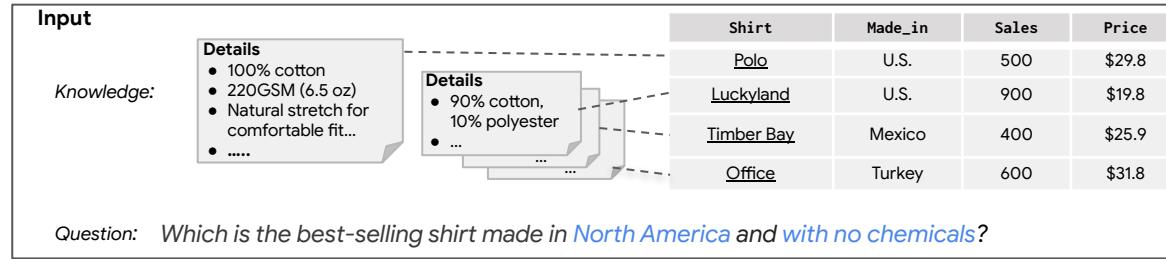
- ✓General

Cons

- ✗Interpretable
- ✗Scalable
- ✗Robust

✗ tool use

LLM + no tool



Pros

- ✓ Improved but still...

Cons

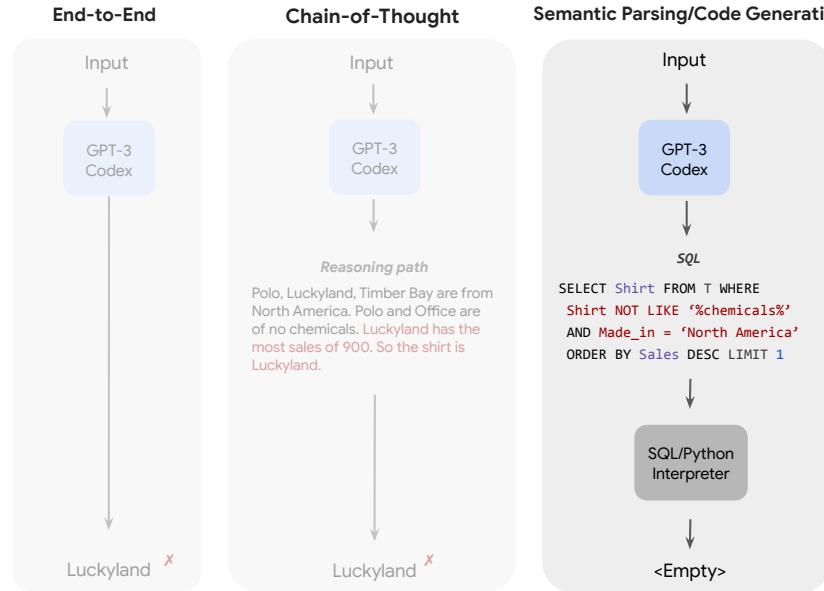
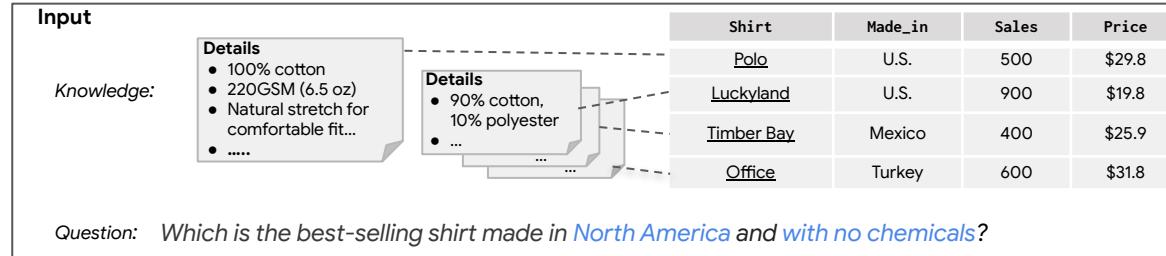
- ✗ Interpretable
- Unreliable answer entailment
- ✗ Scalable
- ✗ Robust

Uninterpretable & not robust

✗ tool use

✗ tool use

LLM + code



Pros

- ✓Interpretable
- ✓Scalable
- ✓Robust

Cons

- ✗Capable

Uninterpretable & not robust

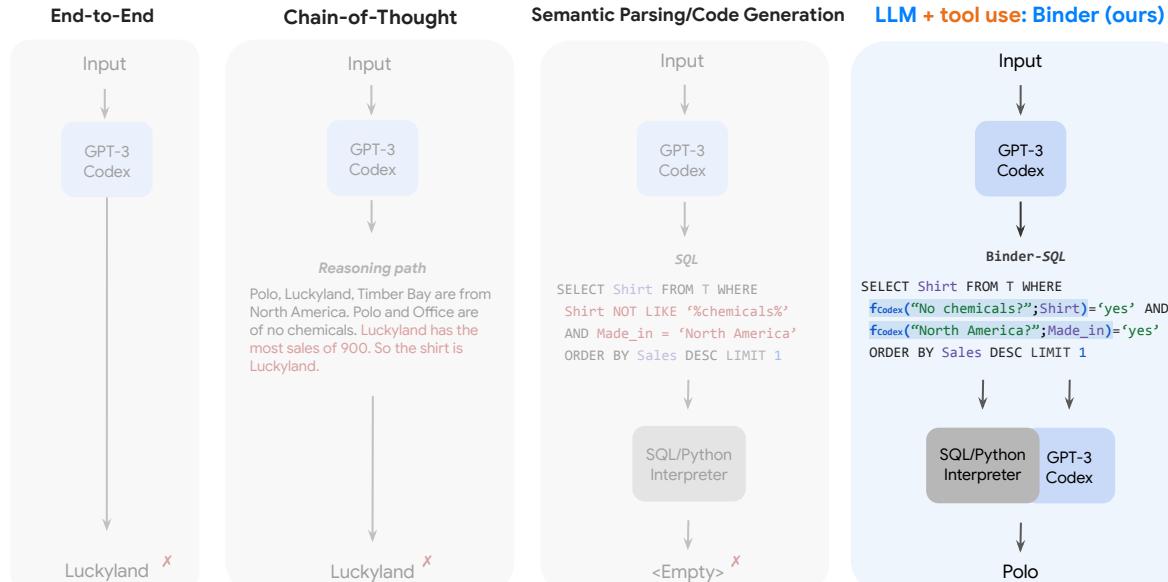
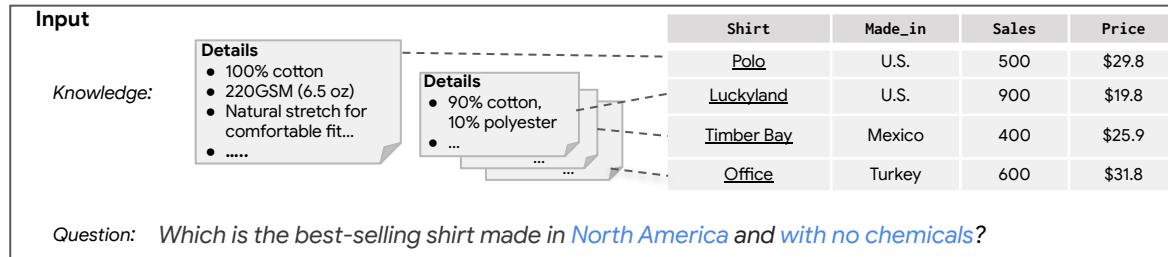
✗ tool use

Unreliable answer entailment

✗ tool use

✓ tool use: SQL/Python

LLM + code and NLP expert function APIs



Why LLM + tool use?

Pros

- ✓Interpretable
- ✓Scalable
- ✓Robust
- ✓Capable

LLM + code: PAL, PoT

Chain-of-Thought (Wei et al., 2022)

Input

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 tennis balls.
balls = 5
2 cans of 3 tennis balls each is
bought_balls = 2 * 3
tennis_balls. The answer is
answer = tennis_balls + bought_balls

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold 93 + 39 = 132 loaves. The grocery store returned 6 loaves. So they had 200 - 132 - 6 = 62 loaves left.

The answer is 62.

X

Program-aided Language models (this work)

Input

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 tennis balls.
balls = 5
2 cans of 3 tennis balls each is
bought_balls = 2 * 3
tennis_balls. The answer is
answer = tennis_balls + bought_balls

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves
loaves_baked = 200
They sold 93 in the morning and 39 in the afternoon
loaves_sold_morning = 93
loaves_sold_afternoon = 39
The grocery store returned 6 loaves.
loaves_returned = 6
The answer is
answer = loaves_baked - loaves_sold_morning
- loaves_sold_afternoon + loaves_returned

>>> print(answer)
74

✓

Question: In Fibonacci sequence, it follows the rule that each number is equal to the sum of the preceding two numbers. Assuming the first two numbers are 0 and 1, what is the 50th number in Fibonacci sequence?

The first number is 0, the second number is 1, therefore, the third number is 0+1=1. The fourth number is 1+1=2. The fifth number is 1+2=3. The sixth number is 2+3=5. The seventh number is 3+5=8. The eighth number is 5+8=13.
.... (Skip 1000 tokens)
The 50th number is 32,432,268,459.

CoT

32,432,268,459

X

```
length_of_fibonacci_sequence = 50
fibonacci_sequence = np.zeros(length_of_)
fibonacci_sequence[0] = 0
fibonacci_sequence[1] = 1
for i in range(3, length_of_fibonacci_sequence):
    fibonacci_sequence[i] = fibonacci_sequence[i-1] +
    fibonacci_sequence[i-2]
ans = fibonacci_sequence[-1]
```

PoT

python

12,586,269,025



LLM + APIs to search/browser for gathering information

Retrieval augmented language models

- Covered in Yuchen and Michi's sessions
- Another ACL tutorial by Akari Asai, Danqi Chen, Sewon Min, Zexuan Zhong

The screenshot shows the landing page for the ACL 2023 Tutorial. At the top, there is a header with the title "WebGPT: Browser-assisted question-answering with human feedback". Below the header, there is a section titled "ACL 2023 Tutorial: Retrieval-based Language Models and Applications". This section features four small profile pictures of the tutorial speakers: Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. Below the speakers' names, it says "Akari Asai¹, Sewon Min¹, Zexuan Zhong², Danqi Chen²". A note at the bottom indicates that ¹University of Washington, ²Princeton University. The date and time of the tutorial are listed as "Sunday July 9 14:00 - 17:30 (EDT) @ Metropolitan West".

The screenshot shows the ChatGPT interface with the model browsing "ALPHA". The user asks, "how do this year's oscar winners compare to recently released movies for box office sales?". The interface displays a search history and results. The search history shows two queries: "Searched: '2023 Oscar winners box office sales compared to recently released movies'" and "Searched: 'recently released movies box office sales'". The results for the first query show a link to "Oscars 2023: Everything Everywhere All at Once Dom..." and a note that it was clicked on. The results for the second query show a link to "Domestic Box Office For Mar 16, 2023 - Box Office Mojo" and a note that it was clicked on. The interface also shows a note that "Reading content" was selected for both results. At the bottom, there is a summary of the results: "The 2023 Oscar winner for Best Picture, "Everything Everywhere All at Once," earned \$100 million at the box office. Recently released movies such as "Scream VI" have grossed \$58,529,284 in its first seven days². While "Scream VI" has had a strong start at the box office, it's worth noting that "Everything Everywhere All at Once" has had a longer run in theaters to accumulate its \$100 million in box office sales."

ChatGPT + browsing

LLM + webs/apps or personalized functions

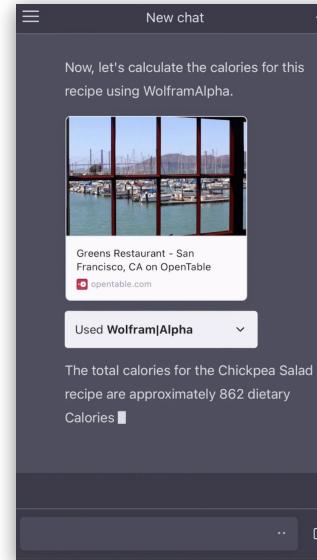
Function calling example

What's the weather like in Boston right now?

Step 1: OpenAI API
Call the model with functions and the user's input

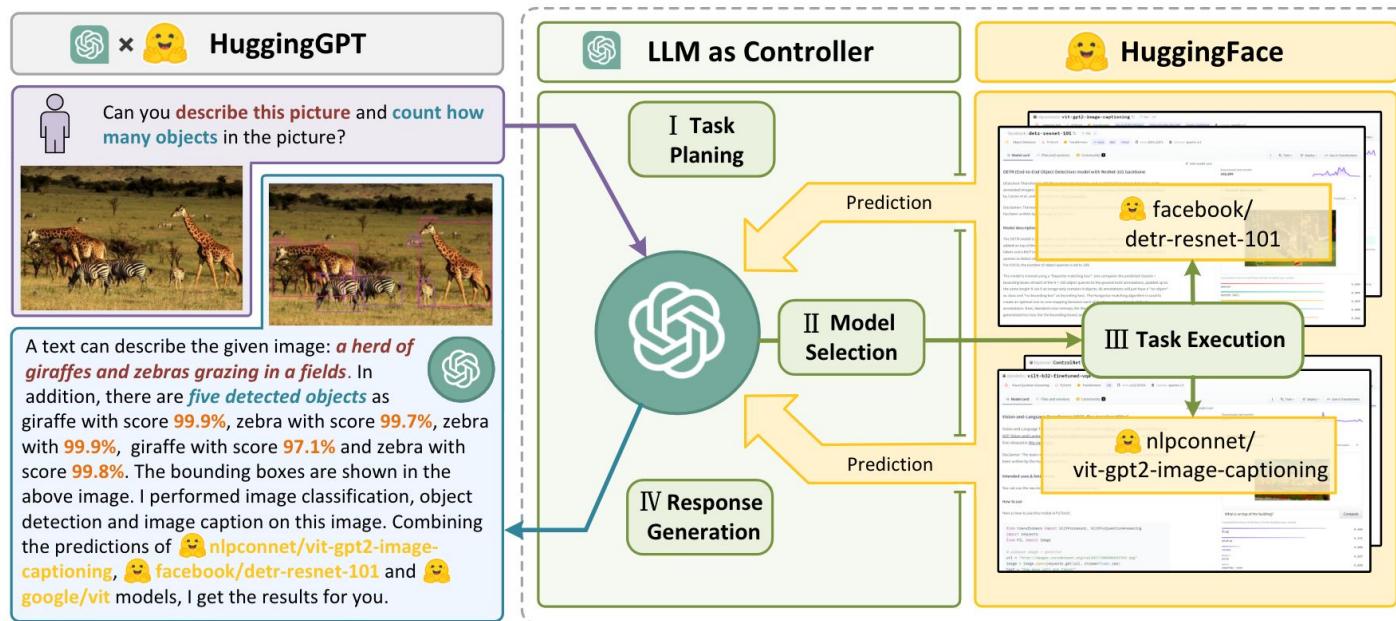
Request Response

```
curl https://api.openai.com/v1/chat/completions -u :$OPENAI_API_KEY -H 'Content-Type: application/json' -d '{ "model": "gpt-3.5-turbo-0613", "messages": [ { "role": "user", "content": "What is the weather like in Boston?" } ], "functions": [ { "name": "get_current_weather", "description": "Get the current weather in a given location", "parameters": { "type": "object", "properties": { "location": { "type": "string", "description": "The city and state, e.g. San Francisco, CA" }, "unit": { "type": "string", "enum": ["celsius", "fahrenheit"] } }, "required": ["location"] } ] }
```



ChatGPT + Plugins: third-party apps/webs, Function calling

LLM + APIs to expert models

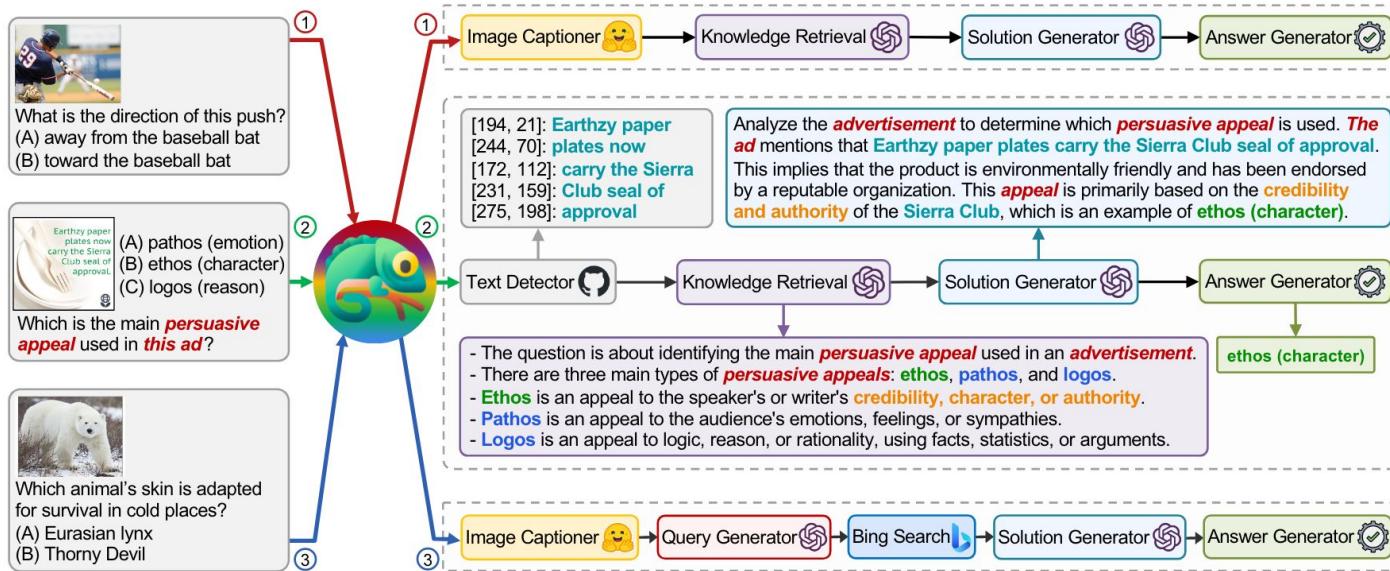


[HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace](#)

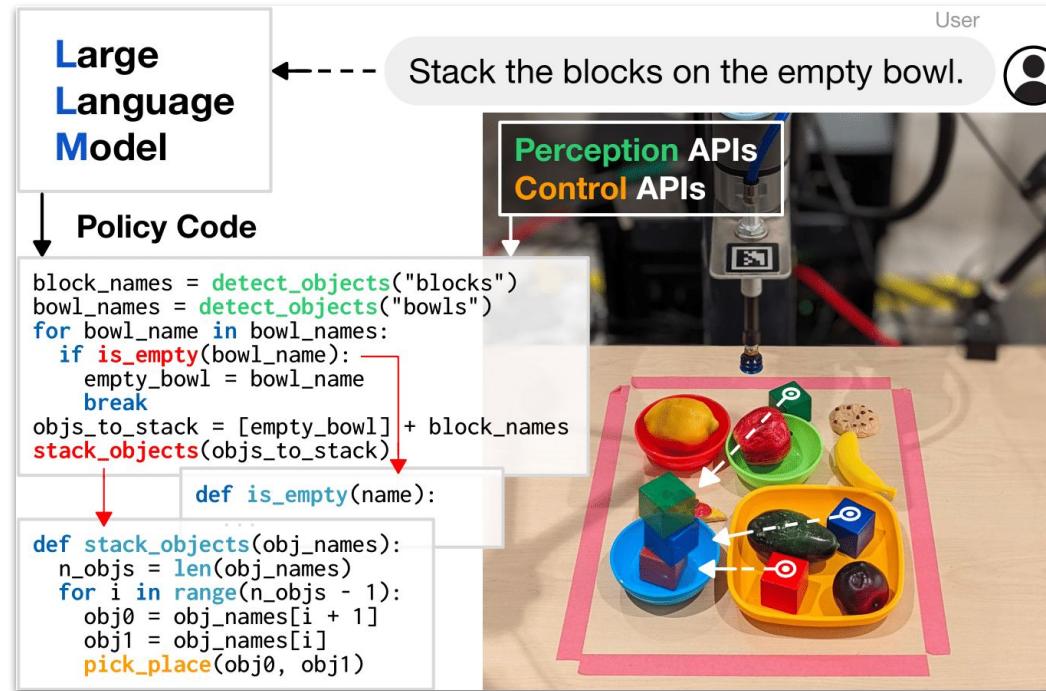
[Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models](#)

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

LLM + APIs to expert models or personalized functions



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



[Code as Policies: Language Model Programs for Embodied Control](#)
[Do As I Can, Not As I Say: Grounding Language in Robotic Affordances](#)
[ProgPrompt: Generating Situated Robot Task Plans using Large Language Models](#)
[Mind's Eye: Grounded Language Model Reasoning through Simulation](#)

LLM + tool use for QA and decision making: ReAct

(2) AlfWorld

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.

Your task is to: Put some pepper shaker on a drawer.

(2a) Act-Only

Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dishponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dishponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens.



(2b) ReAct (Reason + Act)

Act 1: Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.
(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)
Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1, a pepper shaker 1, and a vase 3.
Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.
Act 8: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.
Act 10: Open drawer 1
Obs 10: You open Drawer 1 ...
Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1.



LLM prompting for tool use

Chain-of-Thought (Wei et al., 2022)

Input

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 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold $93 + 39 = 132$ loaves. The grocery store returned 6 loaves. So they had $200 - 132 - 6 = 62$ loaves left.
The answer is 62. X

Program-aided Language models (this work)

Input

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 tennis balls.
`tennis_balls = 5`
2 cans of 3 tennis balls each is
`bought_balls = 2 * 3`
tennis balls. The answer is
`answer = tennis_balls + bought_balls`

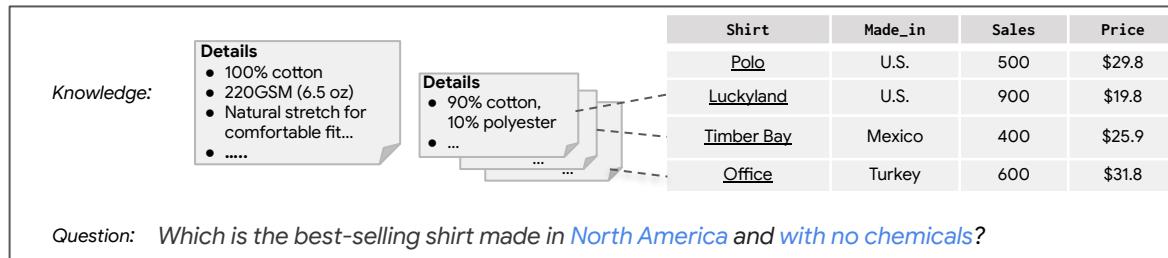
Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

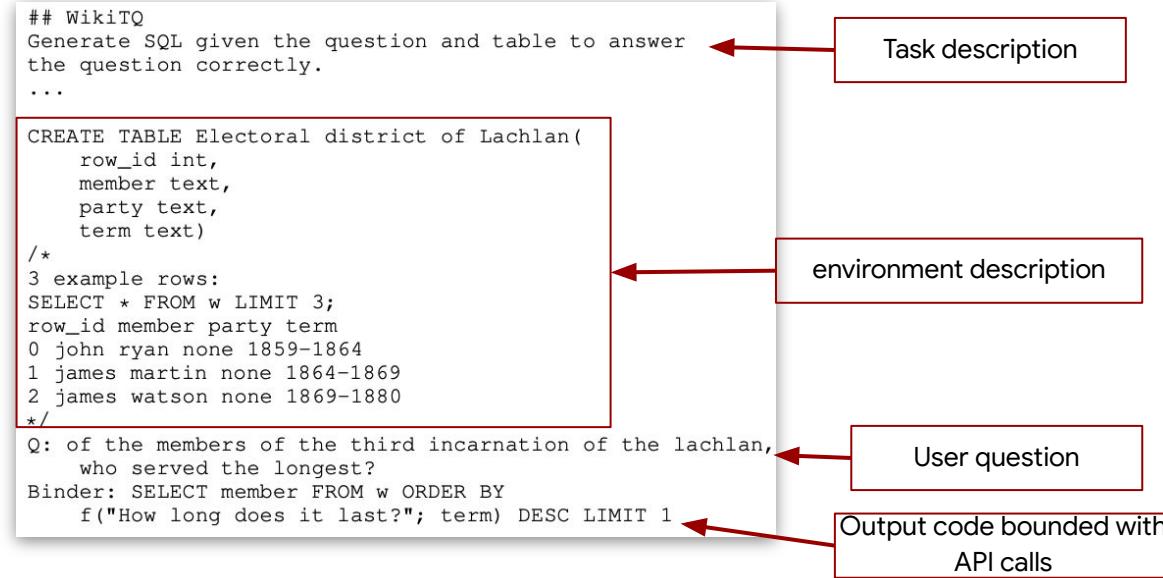
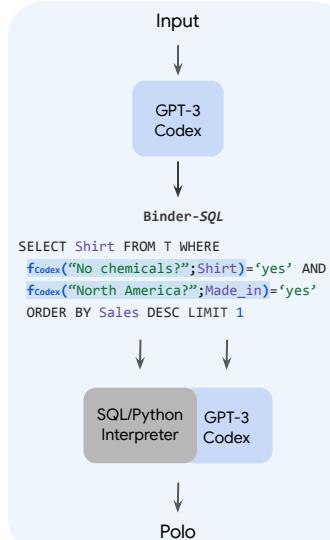
A: The bakers started with 200 loaves
`loaves_baked = 200`
They sold 93 in the morning and 39 in the afternoon
`loaves_sold_morning = 93`
`loaves_sold_afternoon = 39`
The grocery store returned 6 loaves.
`loaves_returned = 6`
The answer is
`answer = loaves_baked - loaves_sold_morning`
`- loaves_sold_afternoon + loaves_returned`

>>> print(answer) ✓

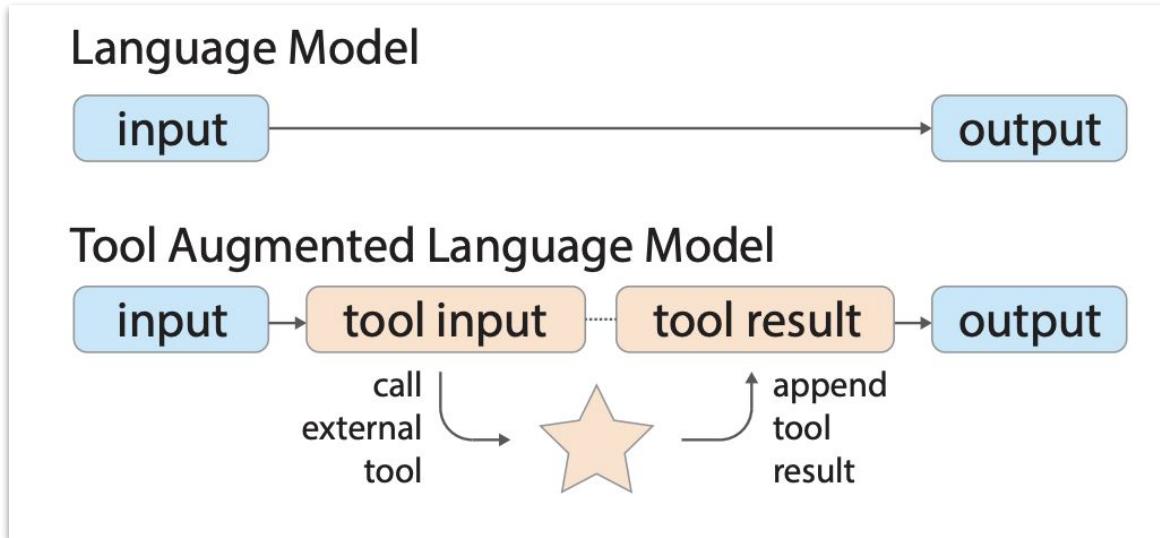
LLM prompting for tool use



LLM + tool use: Binder (ours)



LLM finetuning/pretraining for tool use: TALM



TALM: Tool Augmented Language Models

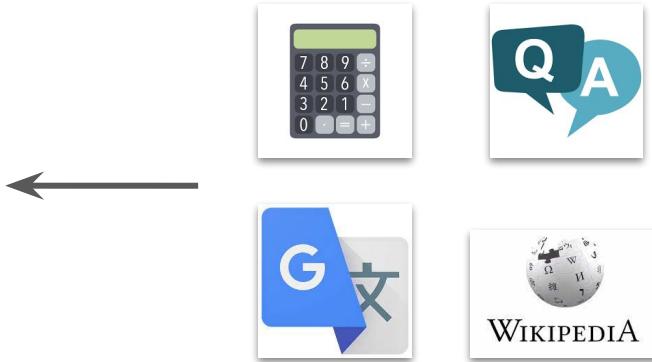
LLM finetuning/pretraining for tool use: Toolformer

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.

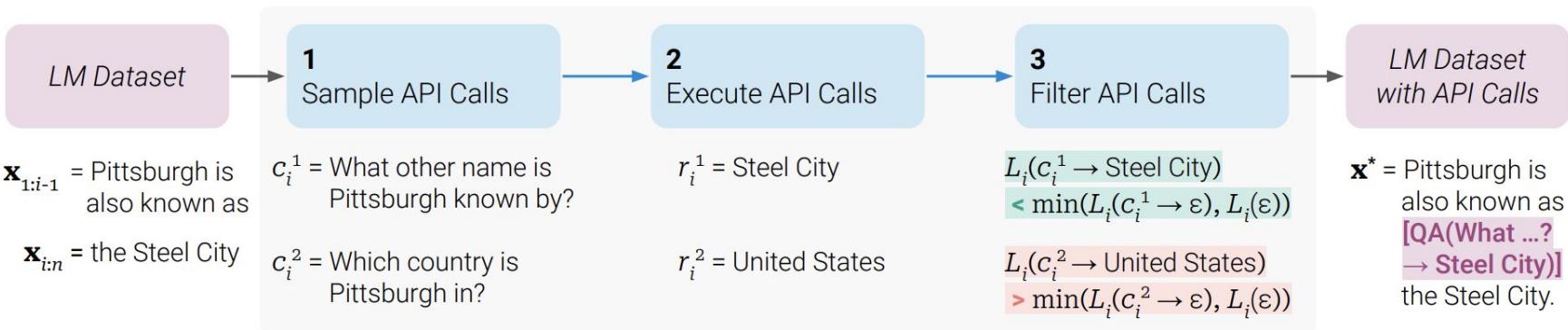
Out of 1400 participants, 400 (or [Calculator($400 / 1400$) → 0.29] 29%) passed the test.

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.



LLM finetuning/pretraining for tool use: Toolformer

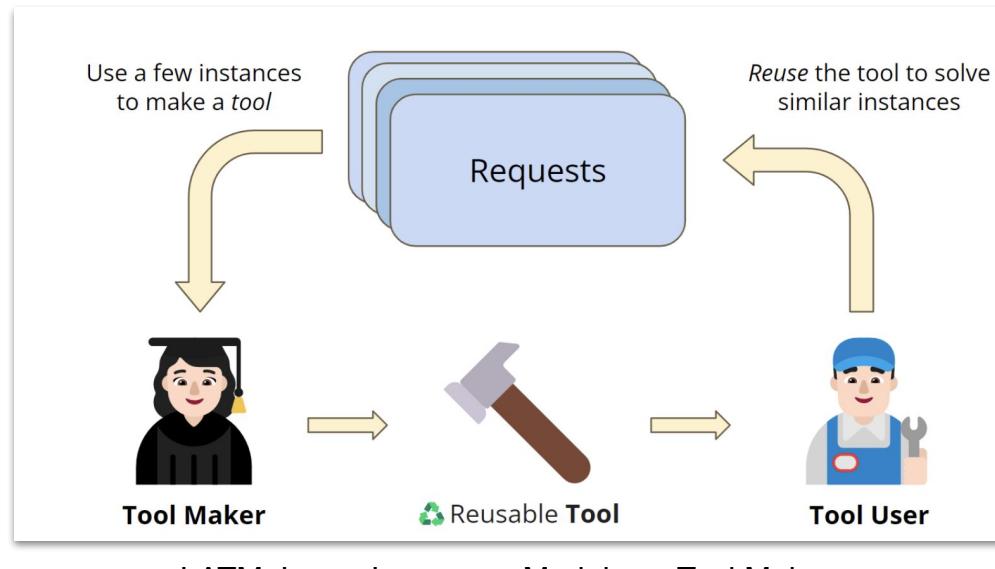


[Toolformer: Language Models Can Teach Themselves to Use Tools](#)

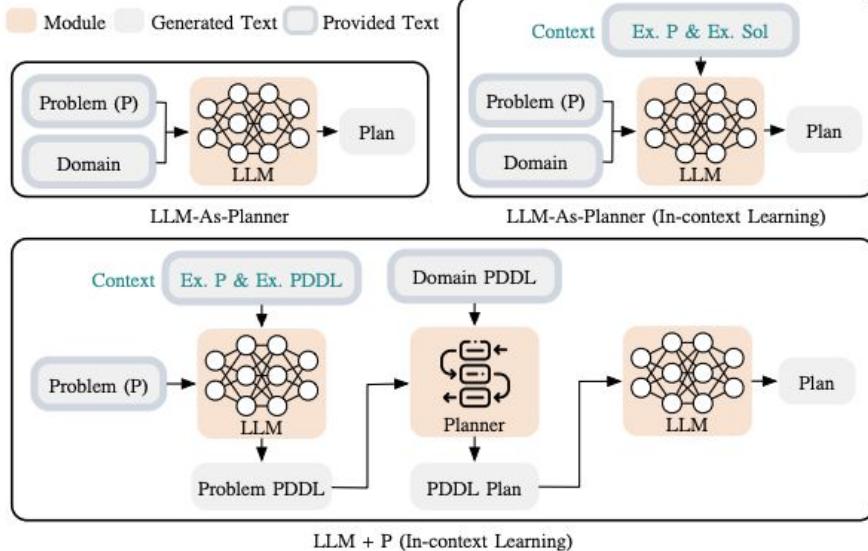
[ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings](#)

[Gorilla: Large Language Model Connected with Massive APIs](#)

Other recent related work: LLM as tool maker



Other recent related work: planning via classical resolver



An Example PDDL Problem File Written by GPT-4 with Context

Context (shaded):

An example planning problem is: You have 5 blocks. b2 is on top of b5. b5 is on top of b1. b1 is on top of b4. b3 is on top of b2. b4 is on the table. b3 is clear. Your arm is empty. Your goal is to move the blocks. b4 should be on top of b3. The problem PDDL is:

```
(:objects b1 b2 b3 b4 b5)
(:init (arm-empty) (on b1 b4) (on b2 b5) (on b3 b2)
      (on-table b4) (on b5 b1) (clear b3))
(:goal (and (on b4 b3))))
```

Prompt: Description of Problem (P1) + Context + Provide me with the problem PDDL file that describes the planning problem directly without further explanations.

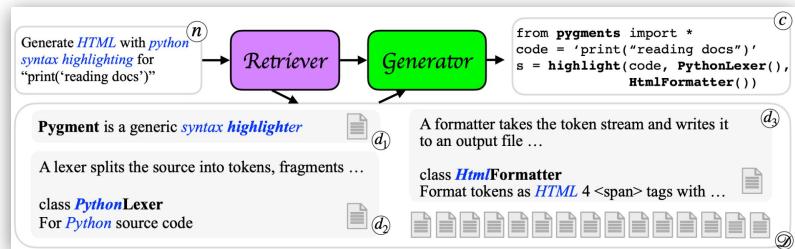
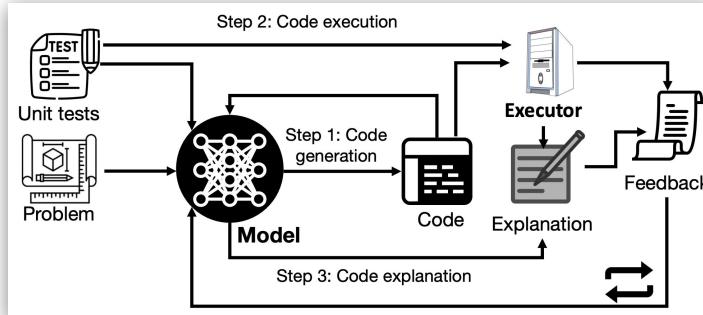
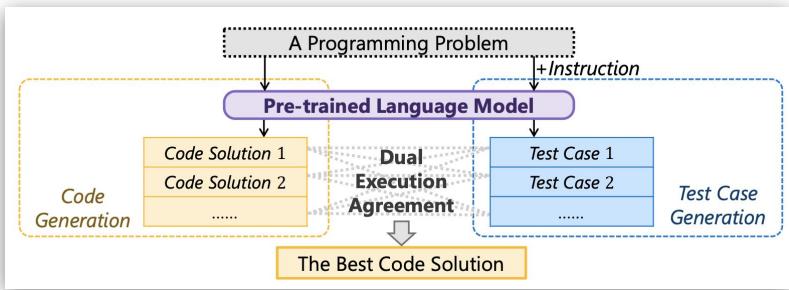
GPT-4 (the generated problem PDDL):

```
(:objects b1 b2 b3 b4 b5 )
(:init (arm-empty) (on-table b1) (on b2 b1) (on b3 b4)
      (on b4 b2) (on b5 b3) (clear b5))
(:goal (and (on b1 b2) (on b3 b5) (on b4 b1)))
```

Planner:

```
(unstack b5 b3) (putdown b5) (unstack b3 b4) (stack b3 b5)
(unstack b4 b2) (putdown b4) (unstack b2 b1) (putdown b2)
(pickup b1) (stack b1 b2) (pickup b4) (stack b4 b1)
```

Other recent related work: code generation



0-shot Python Function Completion

```
import math
def get_decimal(num: float):
    """ return the decimal part
    of the input number
    """

```

Coder: sample program via p(ylx)

```
frac, whole = math.modf(num)  
return frac
```

Reviewer: check program via p(xly)

```
import math
def f(num: float):
    frac, whole = math.modf(num)
    return frac
# write a docstring for the
# above function
def f(num: float):
    """ return the decimal part
    of the input number
    """
```

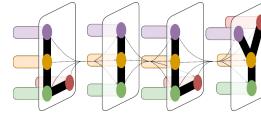
Challenges and future work

- **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?
 -

Thank you!



香港大學自然語言處理實驗室
Natural Language Processing Group, The University of Hong Kong



Any questions?

Thanks to



Google Research

References

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 Model Adaptation 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](#)

Other proof-of-concept projects

- [Auto-GPT](#)
- [LangChain Agent](#)
- [ChatGPT plugins](#)
- [BabyAGI](#)
- [GPT-Engineer](#)
- [ToolBench, BMTools](#)
- ...