

Neuro-Symbolic Approaches

Tutorial on Complex Reasoning over Natural Language
ACL 2023

Language models are strong, what's their lackness?

- Weak symbolic reasoning (for math problem etc.)
- Hallucinations in reasoning (for highly precision-dependency fields)
- Missing / wrong realtime information (for time-sensitive fields)
- Blind real world experience (for robotics etc.)
- No specific / private knowledge (for data-sensitive companies etc.)
- Limited context window length (for structured knowledge grounding etc.)
- ...



Can neuro-symbolic approaches help?

How can we combine neural modules and symbolic components to make our NLP systems more robust and interpretable?

Neuro-symbolic LLMs example: OpenAI Plugins

Model: Browsing ALPHA

how do this year's oscar winners compare to recently released movies for box office sales?

Finished browsing

Searched: "2023 Oscar winners box office sales compared to recently released movies" Clicked on: Oscars 2023: Everything Everywhere All at Once Dom... Reading content Searched: "recently released movies box office sales" Clicked on: Domestic Box Office For Mar 16, 2023 - Box Office Mojo Reading content Finished browsing

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

How many rows does it have? Finished working Show work The "music.csv" dataset has 10,000 rows.

can you run some basic visualizations? Working... Show work Histogram of Song Durations

Stop generating

Model: Plugins ALPHA • Enabled plugins: ⓘ

What human rights were introduced last year?

Used United Nations ⓘ UNVERIFIED

In 2022, one of the human rights that was introduced and emphasized was the human right to a clean, healthy, and sustainable environment. This right was first recognized by the Human Rights Council in 2021, and efforts to realize this right were stepped up in 2022. The recognition of this right was a key recommendation of the Stockholm+50 meeting, where youth and indigenous activists called on governments to respect this human right to a clean, healthy, and sustainable environment (Stockholm, June 2022).

How has thinking on climate change evolved over the years you have access to? Use a table

Using United Nations... ⓘ UNVERIFIED ⓘ

Stop generating

New chat

Now, let's calculate the calories for this recipe using WolframAlpha.

Greens Restaurant - San Francisco, CA on OpenTable

spendable.com

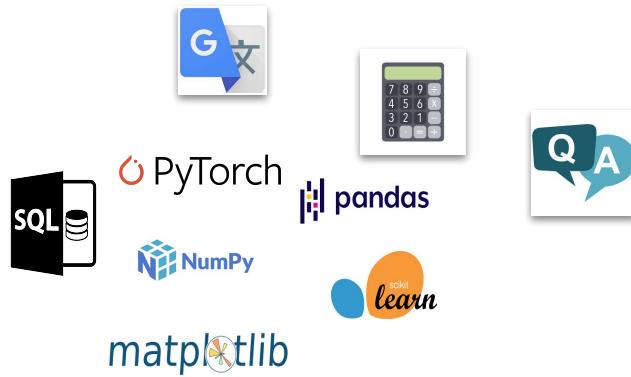
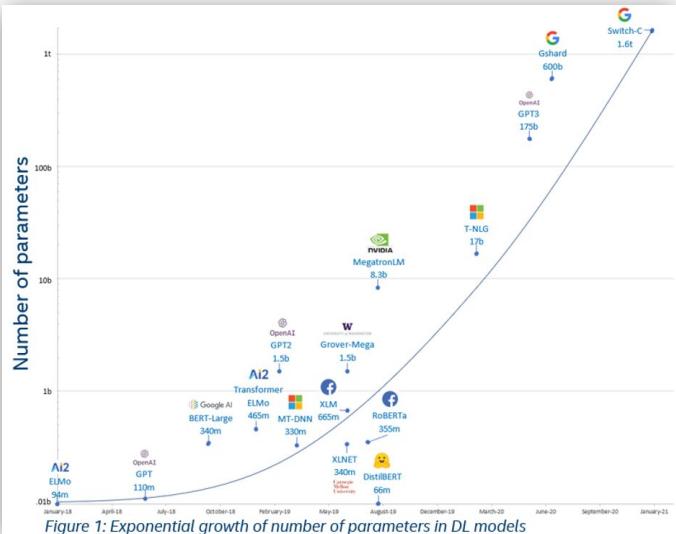
Used Wolfram|Alpha

The total calories for the Chickpea Salad recipe are approximately 862 dietary Calories ■

ChatGPT + Plugins: code interpreter, browsing, retrieval, third-party apps/webs...

Neuro-symbolic focus in this tutorial: LLM + actions/tools/code

In the realm of neuro-symbolic approaches, our main focus lies in empowering **large language models** to utilize diverse tools and plugins (known as “symbolic components” and “neural modules”) to reason.

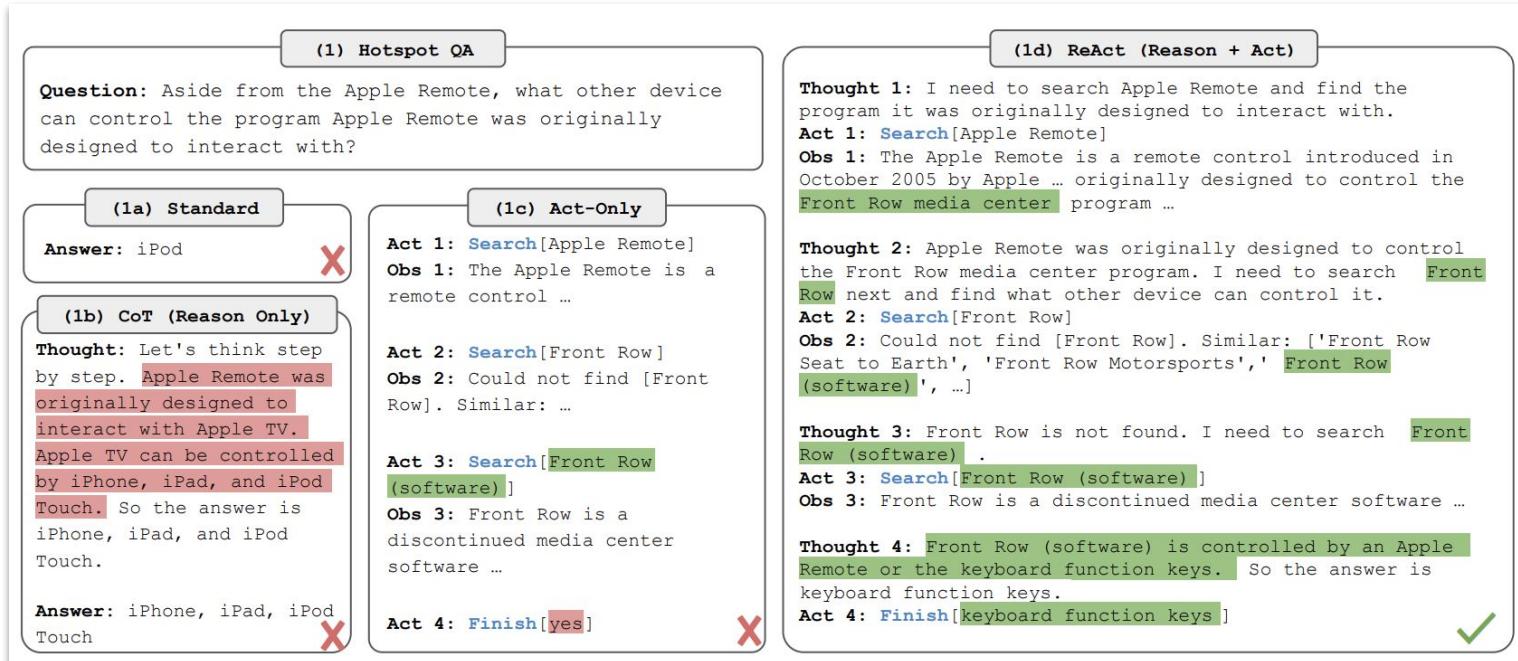


Search Google or type a URL

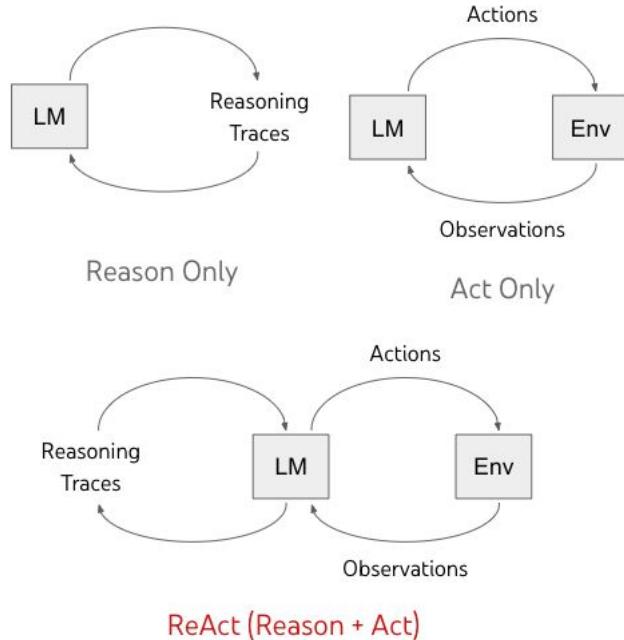
LLM reasoning by leveraging neural modules and symbolic components

- Reasoning by taking actions
- Reasoning by leveraging tools
 - Code Interpreter
 - Neural modules
 - Creating tools
- Reasoning by iterating in environment
 - Through instructions
 - Through code

LLM reasoning by taking pre-defined actions: ReAct

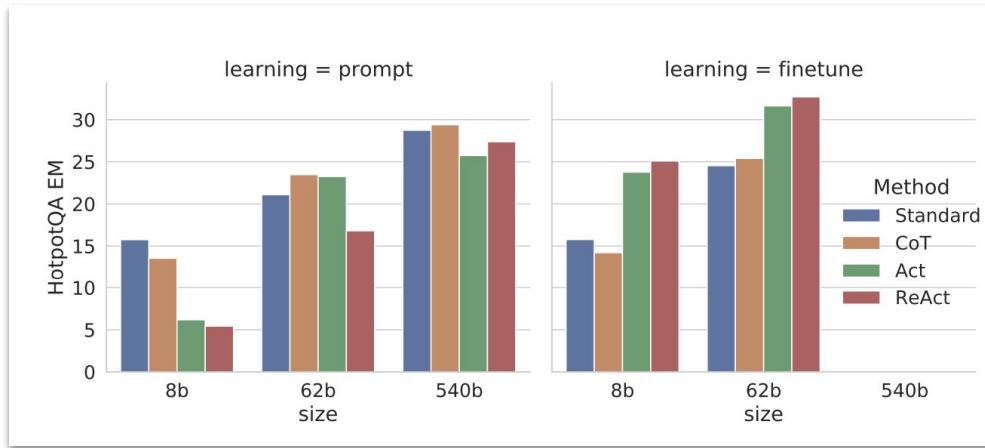


ReAct



- LLMs can be seen as active agents that actively tackle NLP tasks by taking relevant actions.
- Improve the basic interaction paradigm between action and observation by integrating reasoning traces, aiding the reasoning process.
- Become the dominant prompt-based approach for building NLP agents.

Fine-tuning makes better ReAct model than prompting

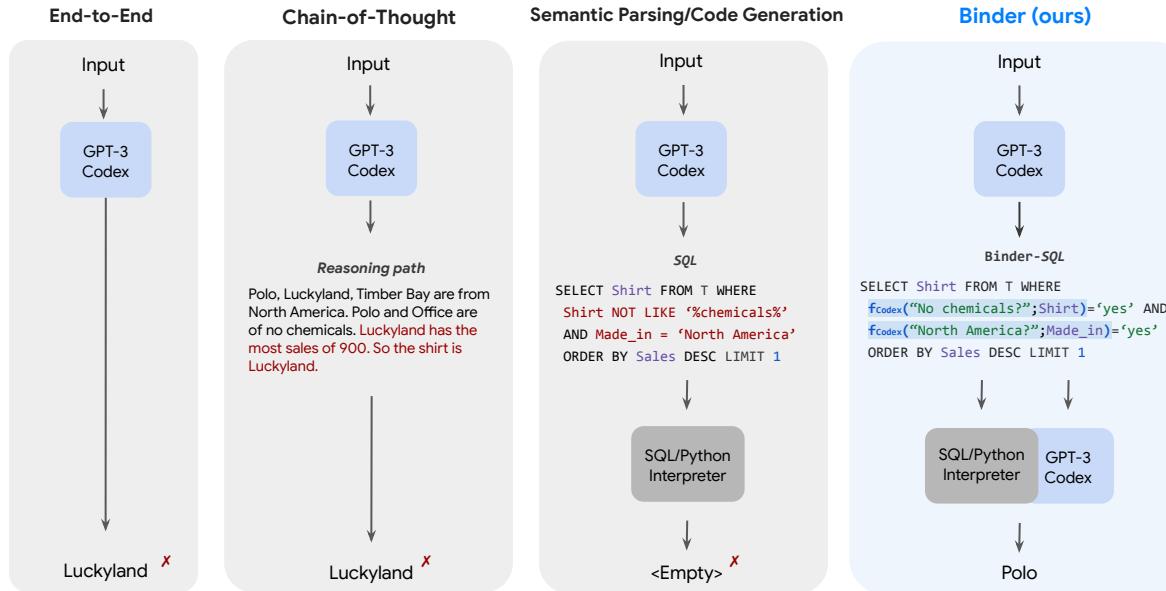
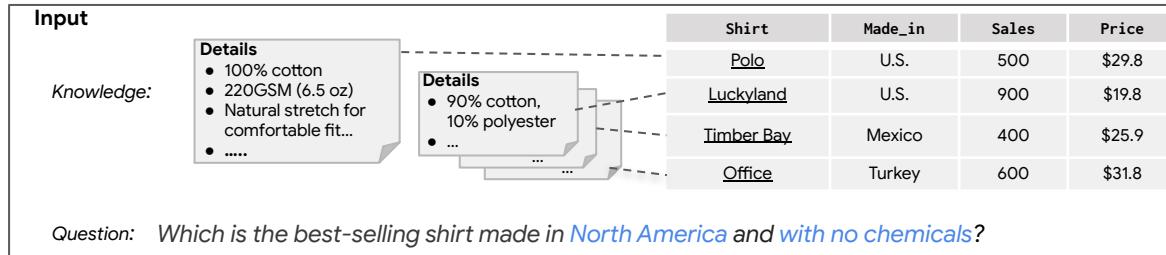


ReAct performs best for fine-tuning

Other works: LLM reasoning by taking actions

- Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022, June). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In International Conference on Machine Learning (pp. 9118–9147). PMLR.
- <https://github.com/hwchase17/langchain/tree/master/langchain/agents>
- <https://openai.com/blog/chatgpt-plugins>
- Qin, Y., Hu, S., Lin, Y., Chen, W., Ding, N., Cui, G., ... & Sun, M. (2023). Tool learning with foundation models. arXiv preprint arXiv:2304.08354.
- Xu, B., Peng, Z., Lei, B., Mukherjee, S., Liu, Y., & Xu, D. (2023). ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models. arXiv preprint arXiv:2305.18323.
- ...

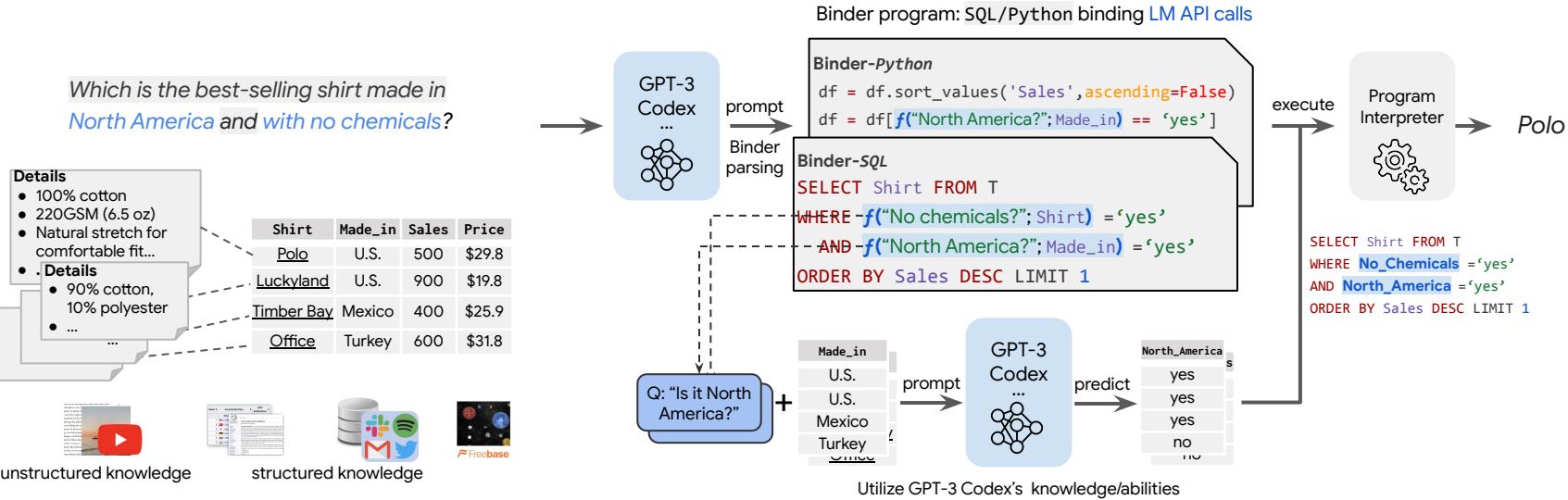
LLM reasoning by leveraging code and neuro modules as tools: Binder



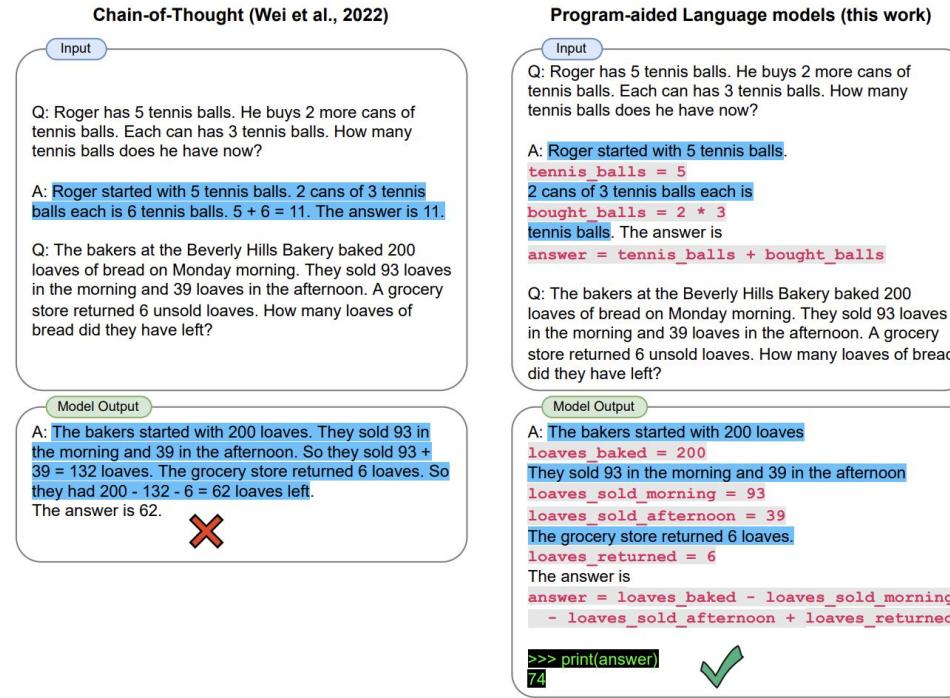
Pros

- ✓Interpretable
- ✓Scalable
- ✓Robust
- ✓Coverage

Binder: Codex+PL interpreter execute → answer



LLM reasoning by leveraging code as tools: PAL



LLM reasoning by leveraging code as tools: PoT

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



```
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



Training models reasoning through 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.

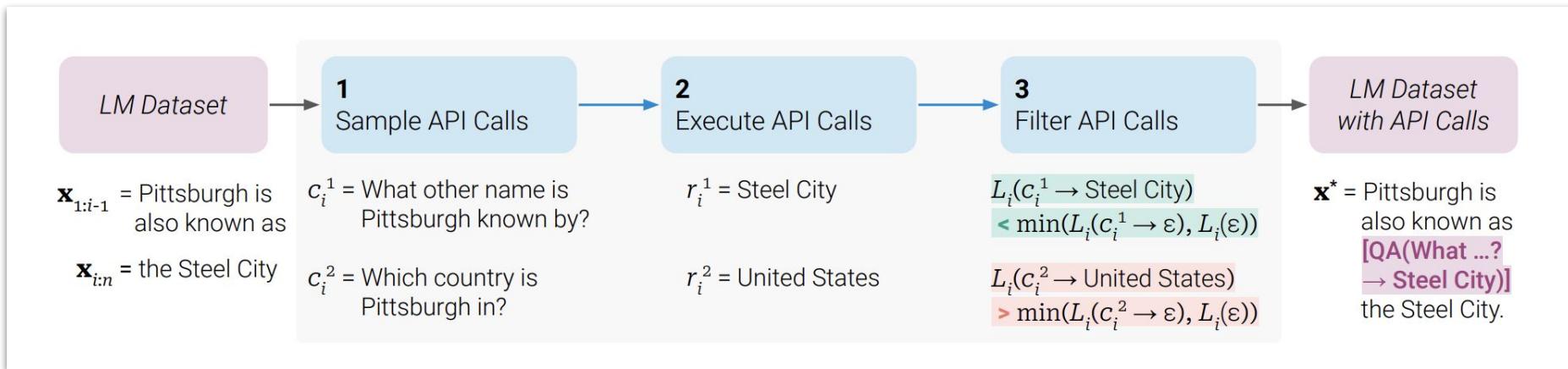
Out of 1400 participants, 400 (or [Calculator($400 / 1400$) → 0.29] 29%) passed the test.

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.



Pipeline for creating tool-use model

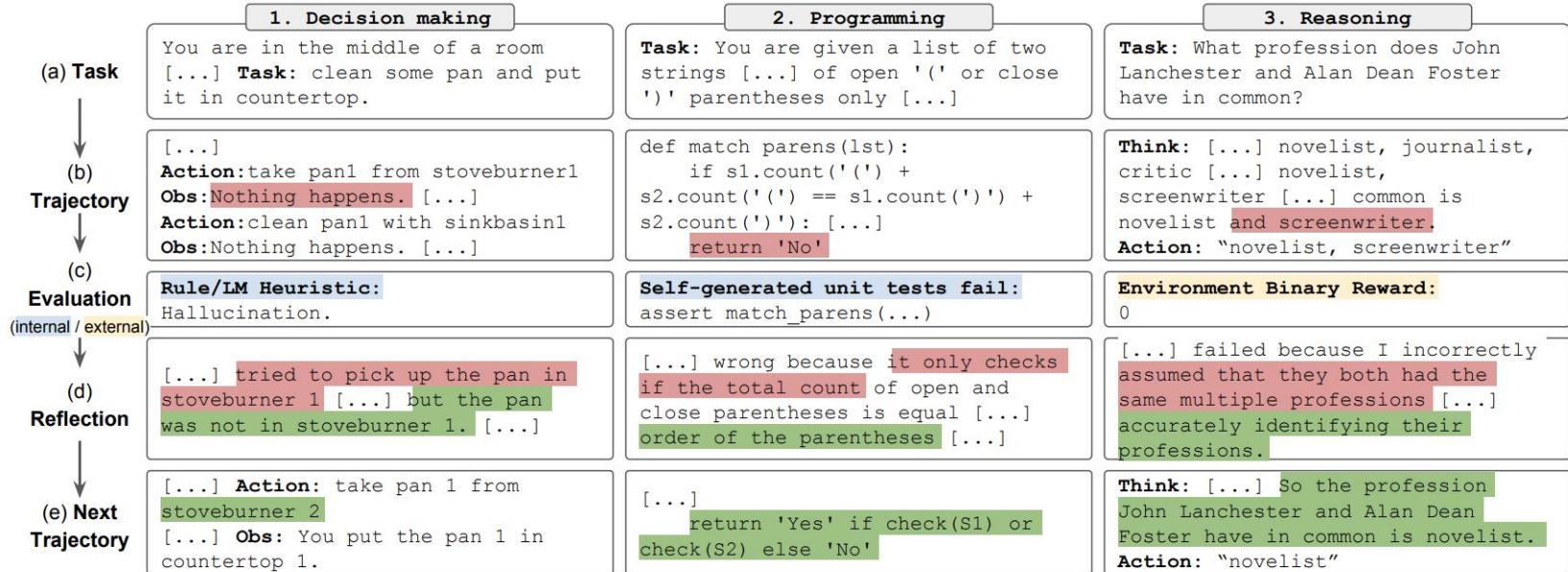


- Propose an efficient approach for leveraging LLMs to acquire and filter data for training a tool-use model.
- Promote the widespread adoption of the tool-use concept.

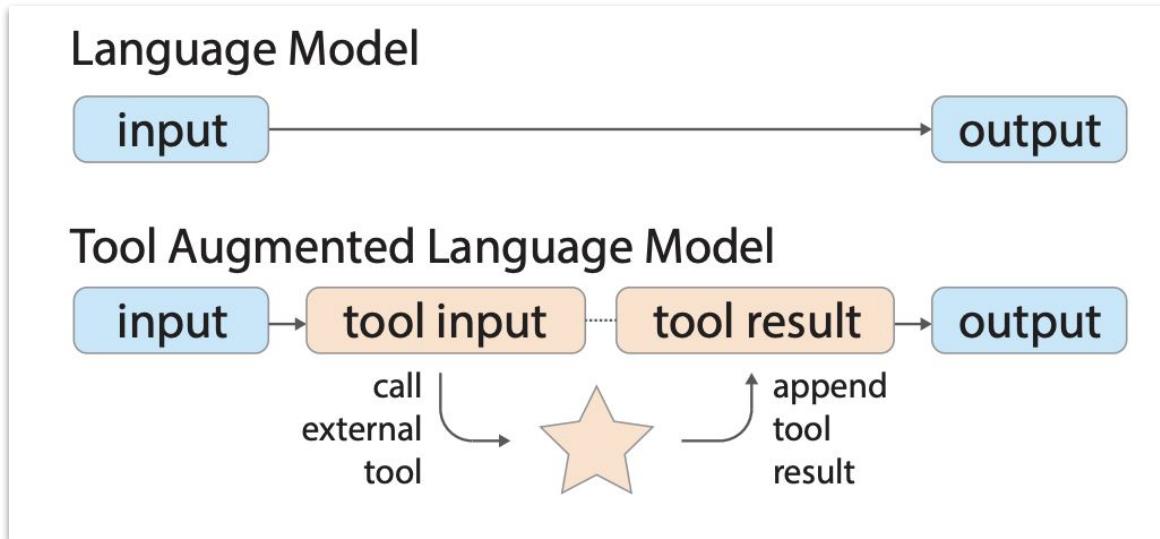
Other works: methods to train models reasoning through tool-use

- <https://github.com/OpenBMB/ToolBench>
- Patil, S. G., Zhang, T., Wang, X., & Gonzalez, J. E. (2023). Gorilla: Large Language Model Connected with Massive APIs. arXiv preprint arXiv:2305.15334.
- Xu, Q., Hong, F., Li, B., Hu, C., Chen, Z., & Zhang, J. (2023). On the Tool Manipulation Capability of Open-source Large Language Models. arXiv preprint arXiv:2305.16504.
- ...

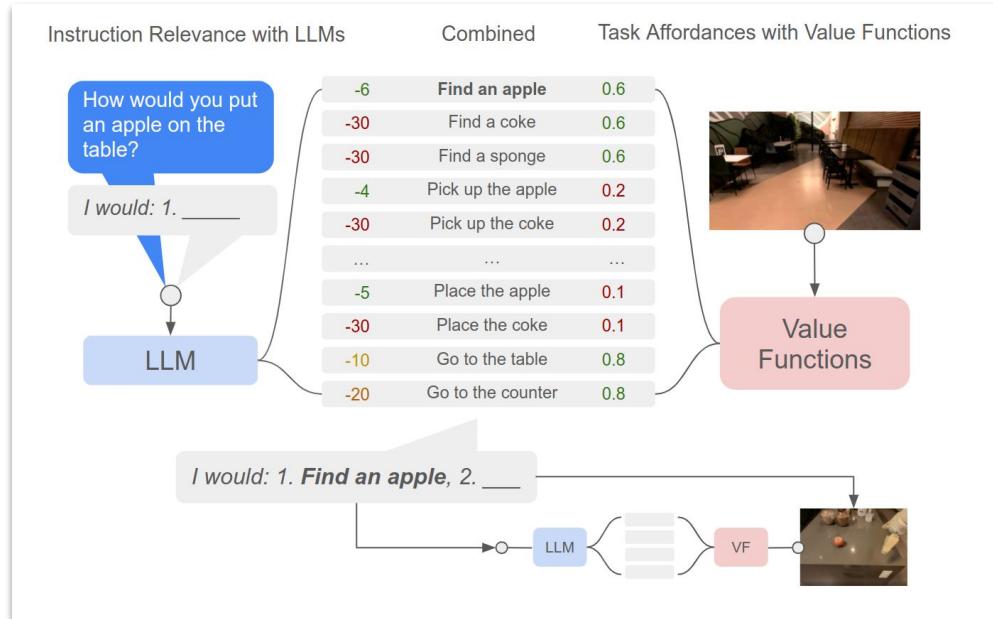
Reasoning with reflexion



TALM: Tool Augmented Language Models

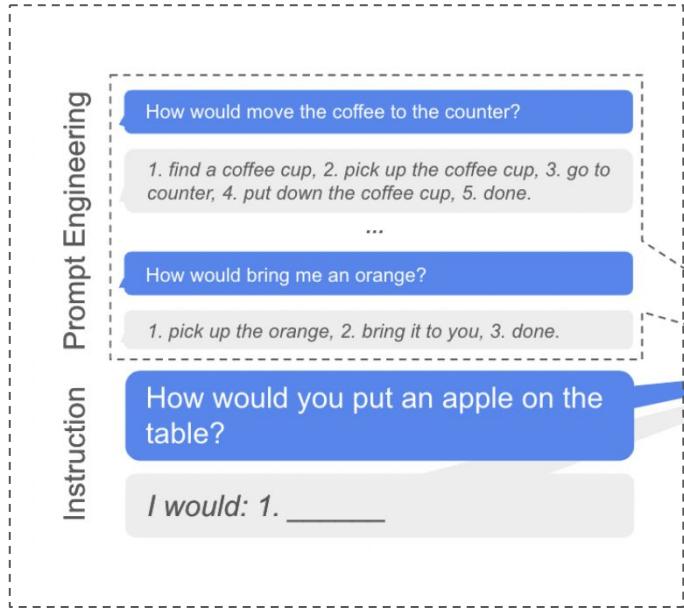


LLM reasoning by iterating in environment (via instructions): SayCan



- Language models **lack real-world experience**, which makes it difficult to leverage them for decision making within a given embodiment.
- Saycan combines the instruction relevance (e.g. Find an apple) from LLMs with the task affordance with Value Functions to make proper choice.

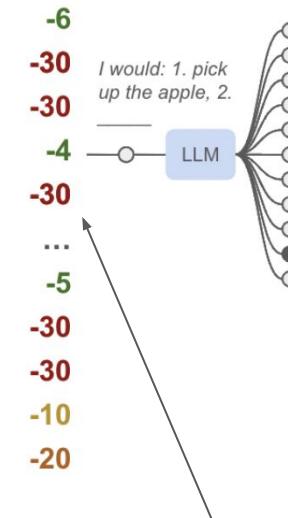
SayCan



In-context learning through prompting

Large Language Model

- Find an apple
- Find a coke
- Find a sponge
- Pick up the apple
- Pick up the coke
- ...
- Place the apple
- Place the coke
- Place the sponge
- Go to the table
- Go to the counter



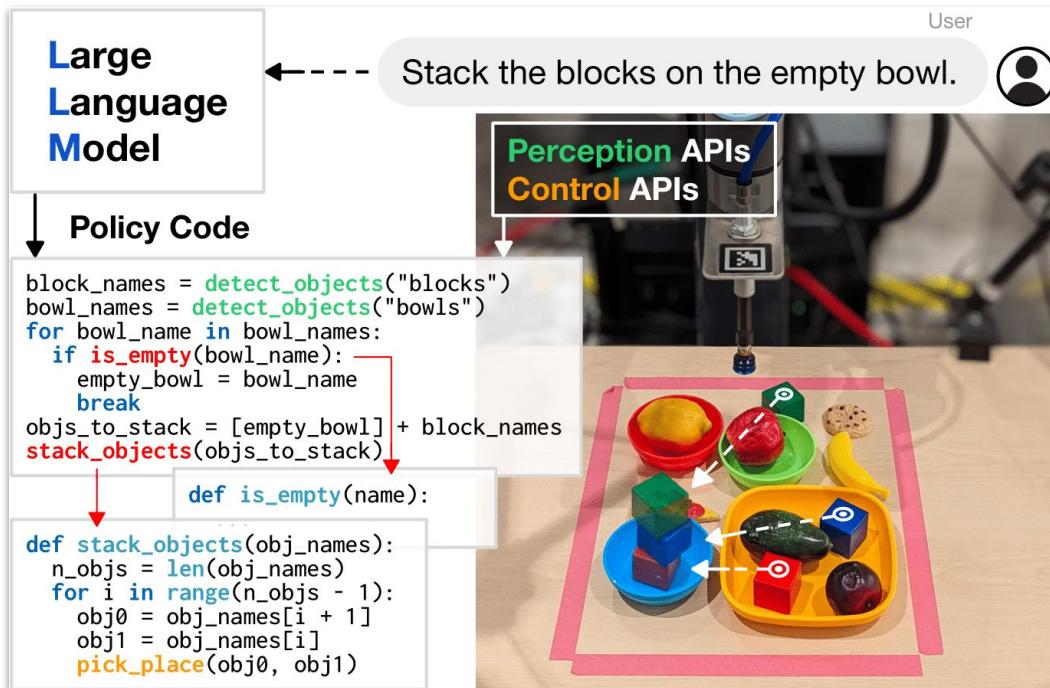
Action affordance through reinforcement learning

SayCan

Other works: LLM reasoning by iterating in environment (via instructions)

- Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022, June). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In International Conference on Machine Learning (pp. 9118–9147). PMLR.
- Zeng, A., Attarian, M., Ichter, B., Choromanski, K., Wong, A., Welker, S., ... & Florence, P. (2022). Socratic models: Composing zero-shot multimodal reasoning with language. arXiv preprint arXiv:2204.00598.
- Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., ... & Ichter, B. (2022). Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608.
- Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W. L., & Su, Y. (2022). Llm-planner: Few-shot grounded planning for embodied agents with large language models. arXiv preprint arXiv:2212.04088.
- ...

LLM reasoning by iterating in environment (via code): Code as Policies



- SayCan is good, although it is constrained to a specific set of natural language instructions and requires iteration to identify the optimal one.
- Conversely, code is deterministic and executable, enabling it to directly control the robot's movements and tasks.

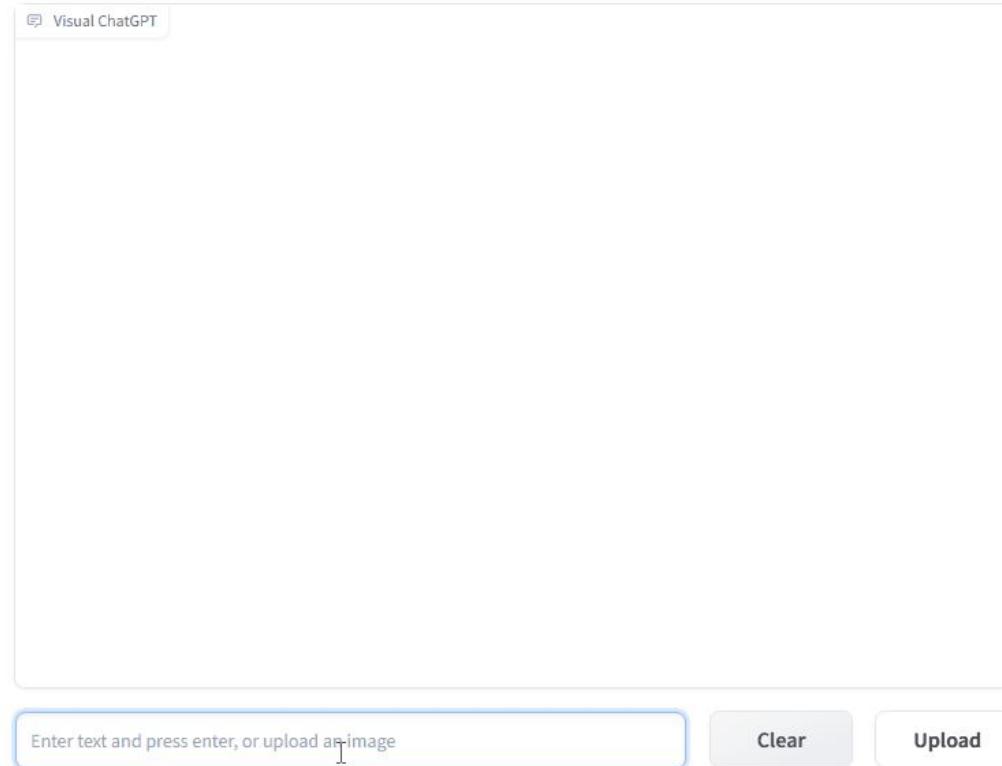
SayCan and Code as Policies

- Starts from researchers from robotic field.
- Done by prompting LLM by in-context examples to make it have the ability to interact with environment, SayCan interacts through constrained set of actions, CaP interacts through code generation.
- Use LLMs as planner. Low level actions are still trained by reinforcement learning / imitation learning and untouched by LLM.

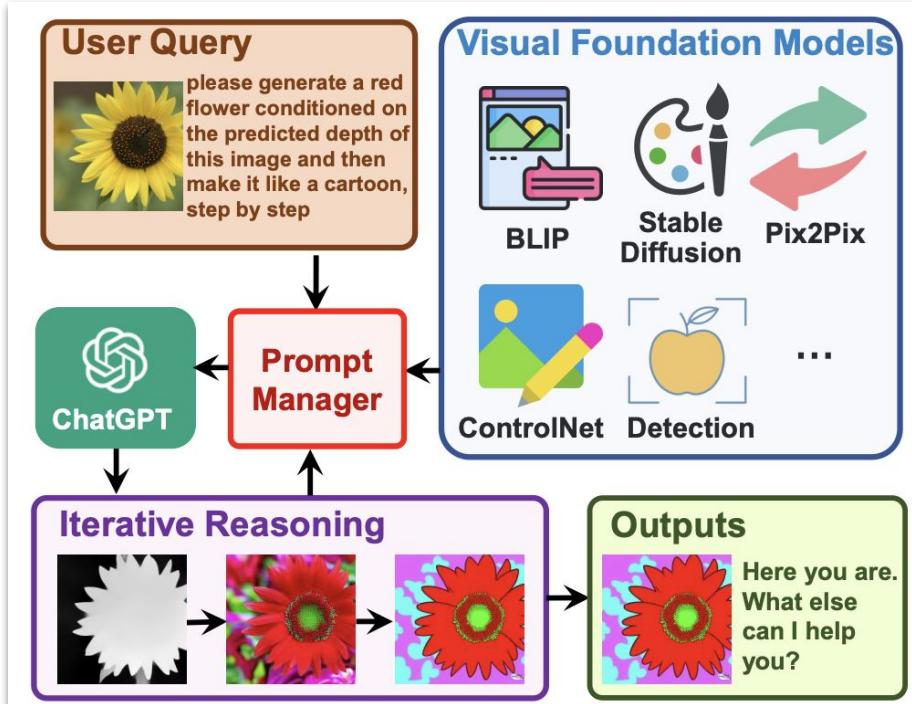
Other works: LLMs reasoning by iterating in environment (via code)

- Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., ... & Garg, A. (2022). Progprompt: Generating situated robot task plans using large language models. arXiv preprint arXiv:2209.11302.
- ...

LLM reasoning by calling neural modules as tools: VisualChatGPT



VisualChatGPT

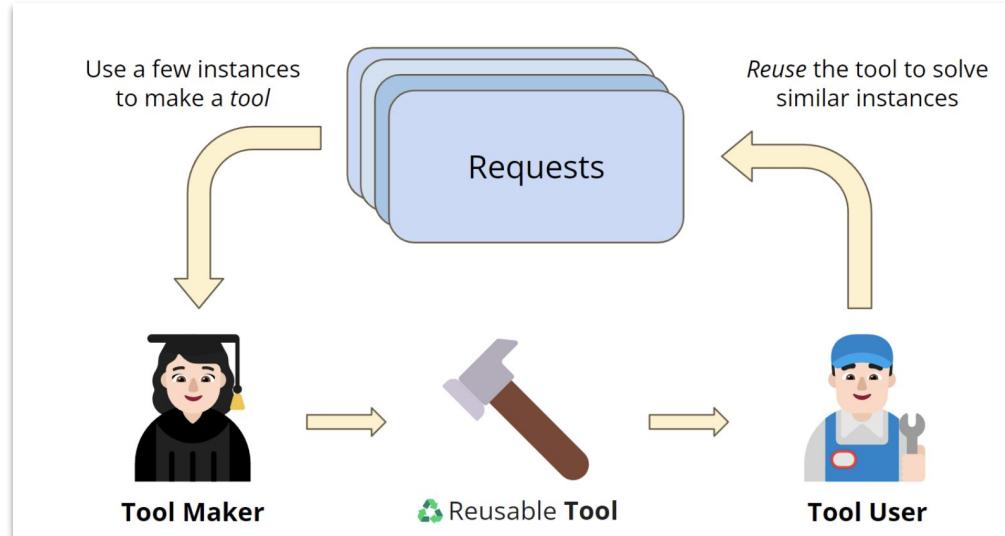


- Enable LLM to query models with natural language interface to pending and receiving **not only languages but also images**.
- Providing complex visual questions or visual editing instructions that require the collaboration of multiple AI models with multi-steps.
- Providing feedback and asking for corrected results.
- Impressive demo system.

Other works: LLM reasoning by calling neural modules as tools

- Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. arXiv preprint arXiv:2303.17580.
- Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K. W., Wu, Y. N., ... & Gao, J. (2023). Chameleon: Plug-and-play compositional reasoning with large language models. arXiv preprint arXiv:2304.09842.
- ...

LATM: Large Language Models as Tool Makers



- LLMs can be also used as a tool-maker.
- Stronger model like GPT-4 can serve as tool-maker, while weaker model as ChatGPT cannot.
- ChatGPT is good enough to be tool-user, and save budget to do so.

Tools can be made if model is strong enough

Tool Maker Model	Logical Deduction (5)	Tracking Shuffled Objects (5)	Dyck Language	Word Sorting	Chinese Remainder Theorem	Schedule Meeting
GPT-3.5 Turbo	0/5	0/5	5/5	5/5	5/5	0/5
GPT-4	3/5	4/5	5/5	5/5	5/5	3/5

Table 3: Success rate of generating new tools (Python functions that pass the tool-verification step) in the tool-making stage with GPT-4 v.s. GPT-3.5 Turbo. We run 5 trials for each model on each task, $n/5$ means n trails out of 5 successes to produce a valid tool. For hard tasks like Logical Deduction and Tracking Shuffled Objects, GPT-3.5 Turbo fails in all trials, showing the necessity of using a more powerful model as *tool maker*.

Other works: LLM reasoning by creating tools

- Qian, C., Han, C., Fung, Y. R., Qin, Y., Liu, Z., & Ji, H. (2023). CREATOR: Disentangling Abstract and Concrete Reasonings of Large Language Models through Tool Creation. arXiv preprint arXiv:2305.14318.
- ...

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](#)

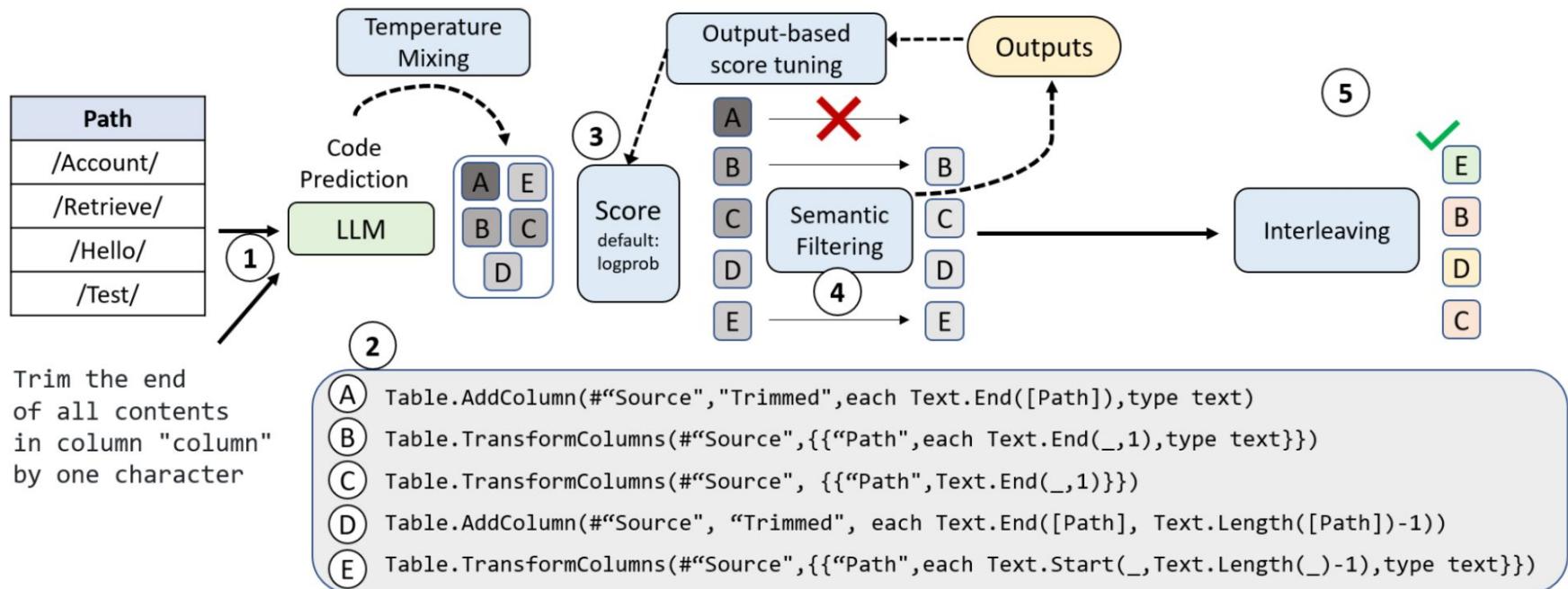
References

- [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](#)

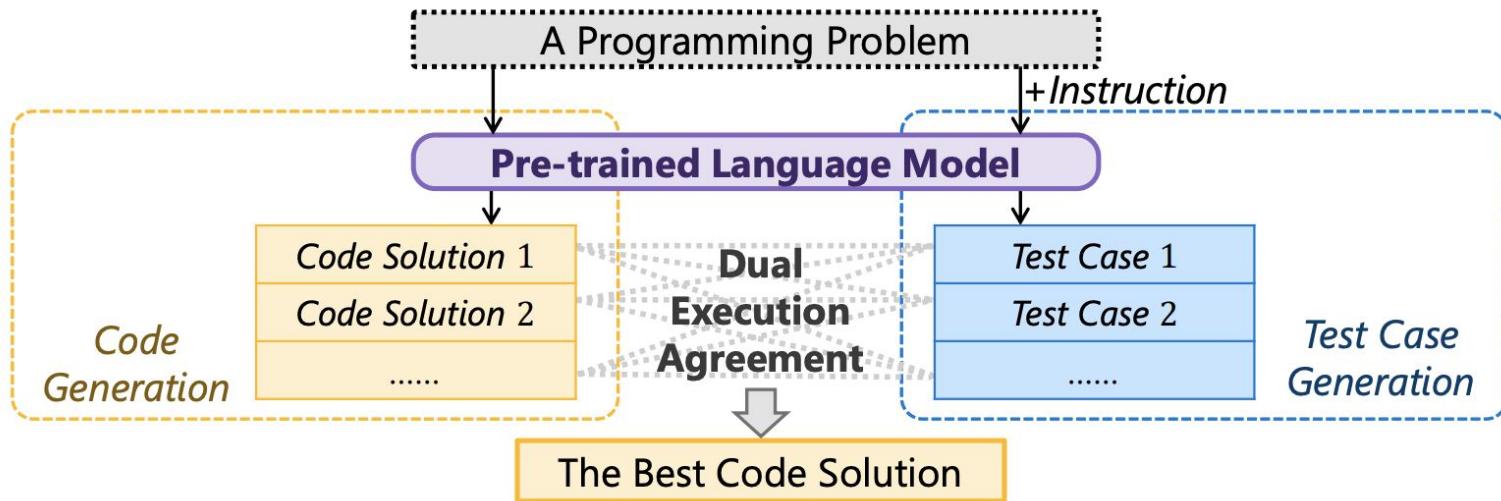
References

- [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](#)

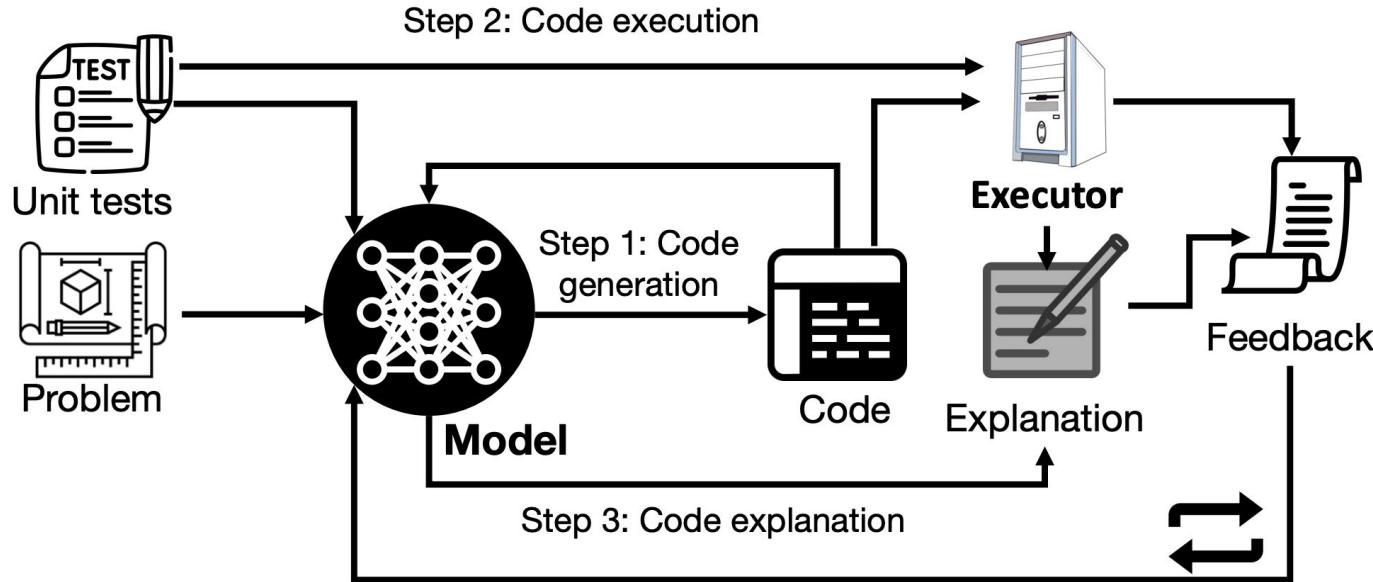
Generate code with data



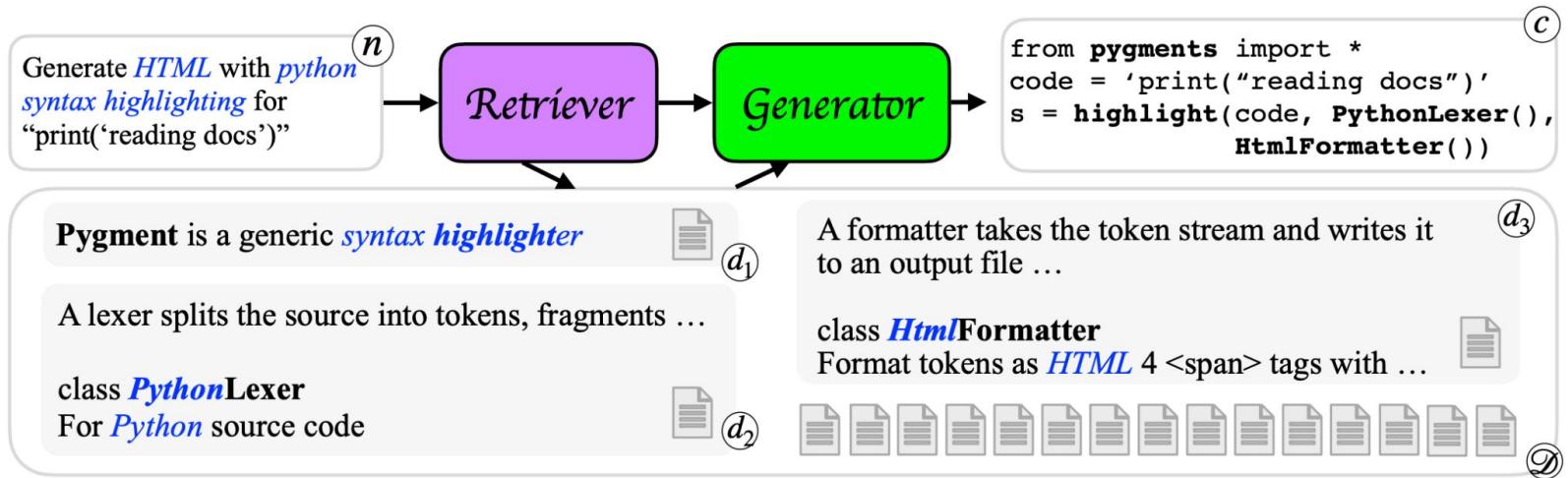
Generate code with test case



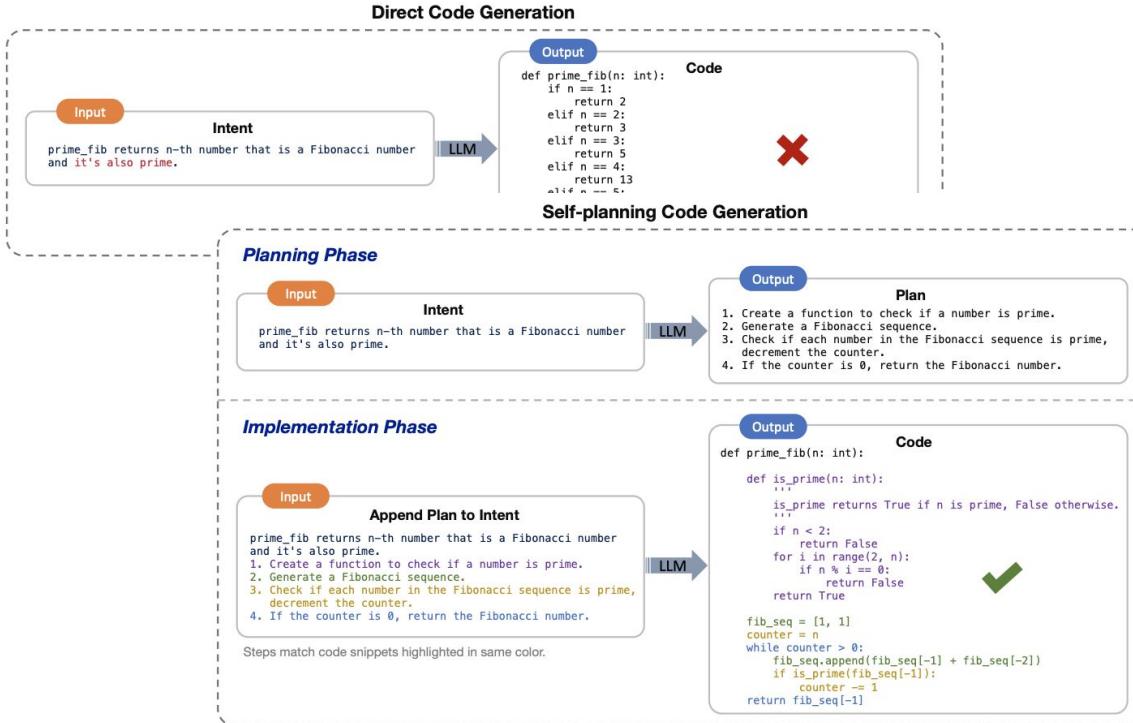
Generate code with explanation



Generate code with documentation



Generate code with planning



Generate code with review

0-shot Python Function Completion

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

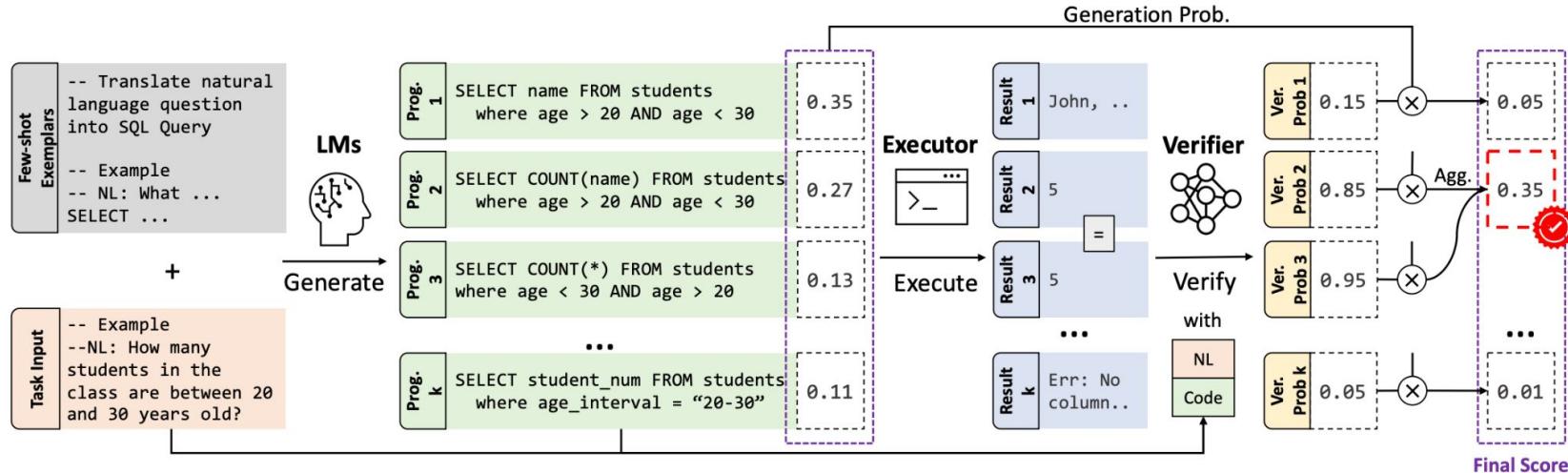
Coder: sample program via p(ylx)

```
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
    """
```

Generate code with verification



Generate code with grammar prompting

LLM Prompt

You are an expert programmer, and you need to write a program for the given natural language query. First, you should write a grammar that contains all the necessary BNF rules. Then, you should write programs that conform to your predicted rules.

(optional) G : [BEGIN RULES] ... [END RULES]

$x^{(1)}$: find the meeting on Wednesday with Bob and Carol

$G[y^{(1)}]$:

```
event      ::= "QueryEvent(" constraint ")"
constraint ::= "(&" constraint constraint ")"
              |
              | "(start_?" date ")"
              |
              | "(attendee_?" attendee attendee ")"
date       ::= "Wednesday"
attendee   ::= "Bob" | "Carol"
```

$y^{(1)}$: QueryEvent(& (start_? Wednesday)(attendee_? Bob Carol))

x : Add meeting with Jean's manager on Wednesday at 3PM

LLM Output

\hat{G} :

```
event      ::= "CreateEvent(" constraint ")"
constraint ::= "(&" constraint constraint ")"
              |
              | "(start_?" date time ")"
              |
              | "(attendee_?" attendee ")"
date       ::= "Wednesday"
time       ::= "NumberPM(3)"
attendee   ::= "FindManager(" attendee ")" | "Jean"
```

\hat{y} : CreateEvent(& (start_? Wednesday NumberPM(3))(attendee_? FindManager(Jean)))

Generate code with evolution

