rao@asu.edu twitter @rao2z @subbarao2z

Can LLMs Reason & Plan?

Subbarao Kambhampati

School of Computing & AI

Arizona State University

Oh by the way, thanks for those Google Faculty Research Awards 2008,2010,2013 & 2016!

Talk on 4/26/2024 @ Google DeepMind

rao@asu.edu @rao2z @subbarao2z

Can LLMs Reason & Plan?

Subbarao Kambhampati
Arizona State University

I come to leverage LLMs, not to lament them..

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z

A clear-eyed understanding of the strengths **and** limitations of a technology is a step towards advancing it.

Blind cheerleading or unalloyed cynicism, in contrast, are just steps towards advancing your influencer career..

Last edited 9:04 AM · Nov 10, 2023 · **25.1K** Views

View post engagements

5 27 118

.. O judgment! thou art fled to brutish beasts,
And men (& LLMs) have lost their reason.

COMMUNICATIONS OF THE ACM

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTI

Home / Blogs / BLOG@CACM / AI as (an Ersatz) Natural Science? / Full Text

BLOG@CACM

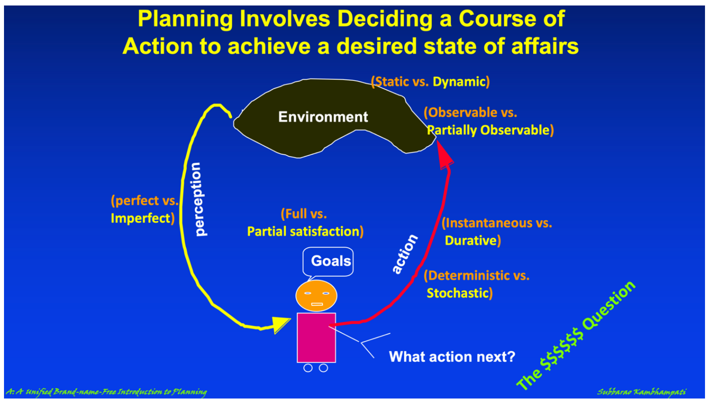
AI as (an Ersatz) Natural Science?

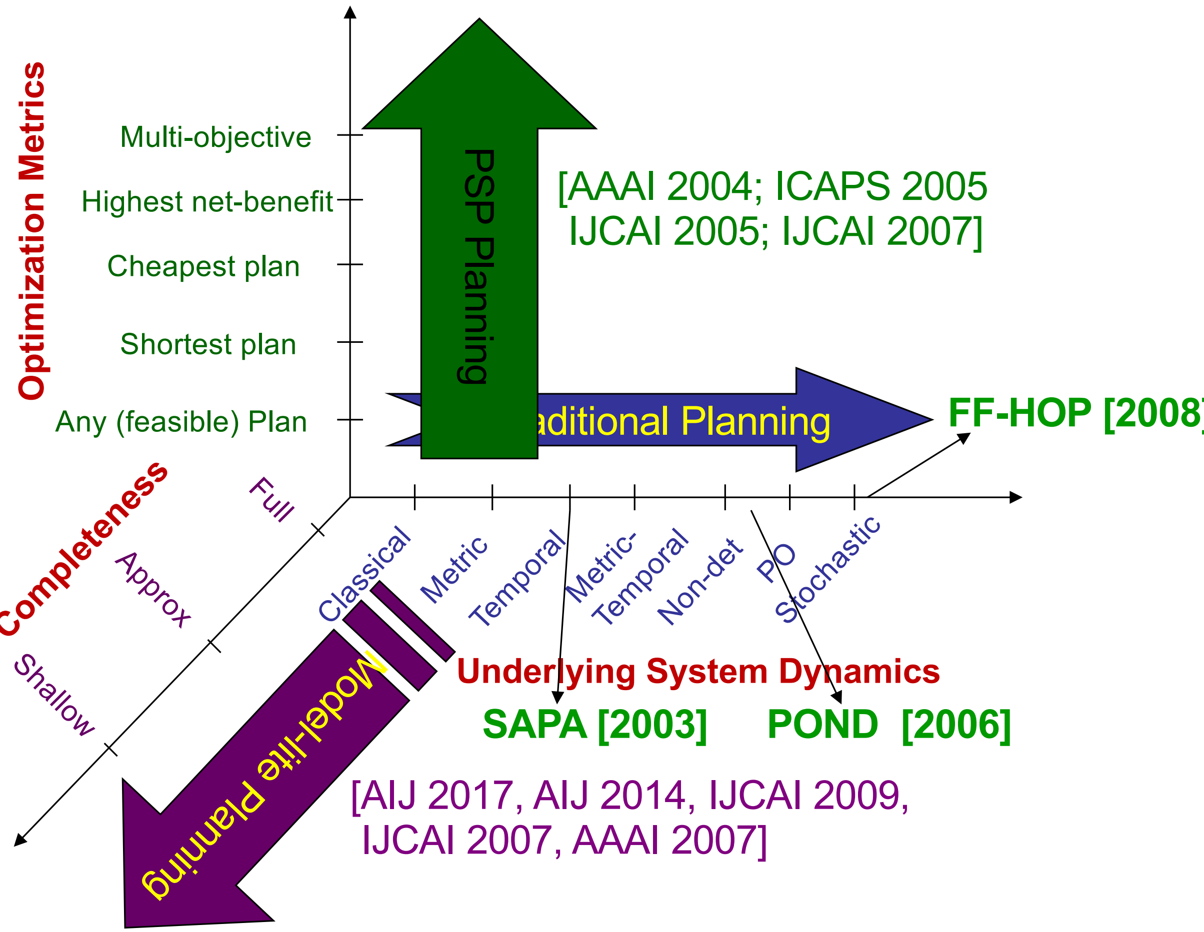
By Subbarao Kambhampati
June 8, 2022
Comments (1)

VIEW AS: SHARE:

In many ways, we are living in quite a wondrous time for AI, with every week bringing some awe-inspiring feat in yet another tacit knowledge task that we were sure would be out of reach of computers for quite some time to come. Of particular recent interest are the large learned systems based on transformer architectures that are trained with billions of parameters over massive Web-scale multimodal corpora. Prominent examples include large language models like GPT3 and PALM that respond to free-form text prompts, and language/image models like DALL-E and Imagen that can map text prompts to photorealistic images (and even those with claims to general behaviors such as GATO) .

The emergence of these large learned models is also changing the nature of AI research in fundamental ways. Just the other day, some researchers were playing with DALL-E and thought that it seems to have developed a secret language of its own which, if we can master, might allow us to interact with it better. Other researchers found that GPT3's responses to reasoning questions can be improved by adding certain seemingly magical incantations to the



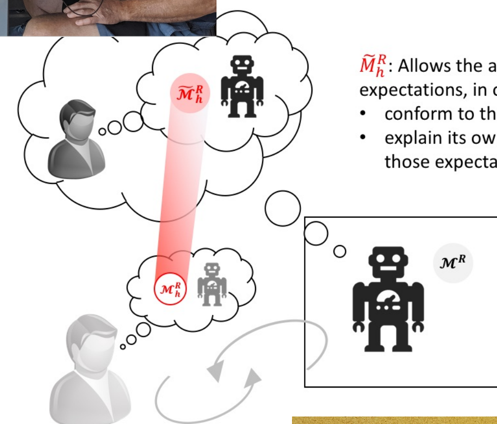


Information Gathering;
Information Integration

RL with Simulator is Planning..

Human-AI Interaction

- We have focused on explainable human-AI interaction.
- Our setting involves collaborative problem solving, where the AI agents provide decision support to the human users in the context of *explicit knowledge sequential decision-making tasks* (such as mission planning)
    - In contrast, much work in social robotics and HRI has focused on tacit knowledge tasks (thus making explanations mostly moot)
    - We assume that the AI agent either learns the human model or has prior access to it.
- We have developed frameworks for proactive explanations based on *model reconciliation* as well as on-demand *foil-based explanations*
- We have demonstrated the effectiveness of our techniques with systematic (IRB approved) human subject studies

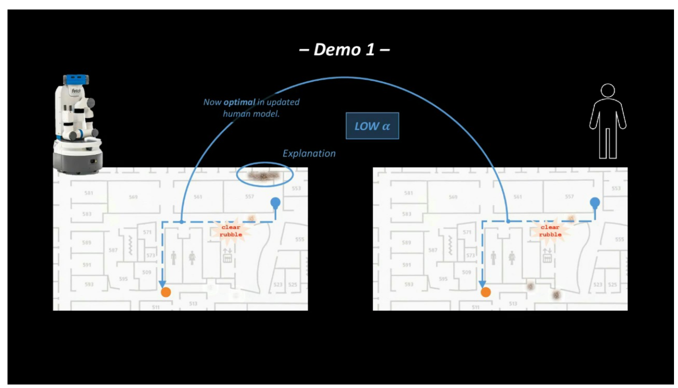


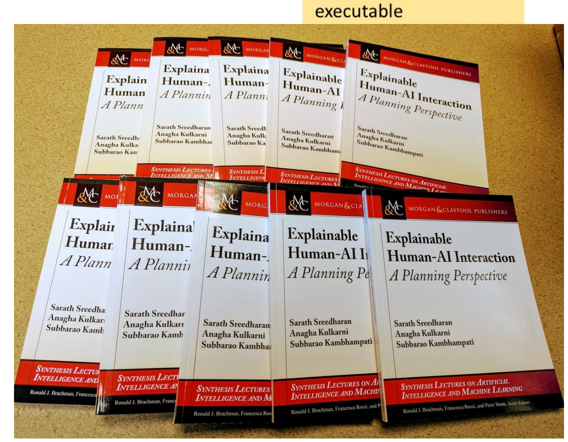
$\widetilde{M}_h^R$: Allows the agent to anticipate human expectations, in order to

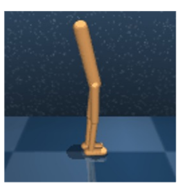
- conform to those expectations
- explain its own behavior in terms of those expectations.

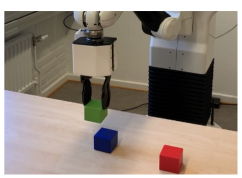
M_r^H and $\widetilde{M}_h^R$ are *Expectations* on Models $\mathcal{M}^H$ and $\mathcal{M}^R$

They don't have to be executable









COMMUNICATIONS OF THE ACM

BLOG@CACM

Artificial Intelligence and Machine Learning

Language Imitation Games and the Arrival of Broad and Shallow AI

By Subbarao Kambhampati

Posted Oct 7 2021

When my son was still a toddler and his mom had to go on an extended trip out of the country, he would "talk" to her on the phone almost daily. Scare quotes because he still was more babbling than talking. But, the impressive (and adorable) thing was that his imitation of the syntactics of us talking on the phone was flawless, replete with the meaningful

The New York Times

Quotation of the Day: When Chatbots 'Hallucinate'

May 8, 2023

"If you don't know an answer to a question already, I would not give the question to one of these systems."

SUBBARAO KAMBHAMPATI, a professor and researcher of artificial intelligence at Arizona State University, about inaccuracies in information from chatbots, sometimes referred to as "hallucinations."

THE HILL

THE VIEWS EXPRESSED BY CONTRIBUTORS ARE THEIR OWN AND NOT THE VIEW OF THE HILL

Beauty, lies & ChatGPT: Welcome to the post-truth world

BY SUBBARAO KAMBHAMPATI, OPINION CONTRIBUTOR - 02/16/23 10:00 AM ET

AP Photo/Timothy D. Easley

Bella Whitice talks with classmate Katherine McCormick as they try and outwit the "robot" that was creating writing assignments in Donnie Piercey's class at Stonewall Elementary in Lexington, Ky., Monday, Feb. 6, 2023. The robot was the new artificial intelligence tool ChatGPT which can generate everything from essays and haikus to term papers in a matter of seconds.

Two months back, a company called OpenAI released its chatbot, ChatGPT, to the public. ChatGPT is a so-called Large Language Model (LLM) that is trained on the nearly 600 gigabytes of text of all kinds found on the World Wide Web to learn to complete any text prompt by predicting the next word, and the word after that, and so on. The purported aim of the system is to put the "auto complete" functionality for words, found on cellphones, on steroids so it can complete entire paragraphs.

The powers of these types of LLMs have long been known in the technology sector, thanks to ChatGPT's predecessor, GPT3, from OpenAI and similar systems from other Big Tech companies.

Of course, human memory is itself not veridical — we don't store and retrieve experiences verbatim but, instead, stitch them on demand (thus leading to false memories and unreliable witnesses). However, unlike humans who can (sometimes) verify their memories against external sources, LLMs focus just on the statistical likelihood of the completion provided. They do not have any model of the world we inhabit beyond this.

Thus, in the case of ChatGPT, all meaning and accuracy — beyond plausible completion in the context of training data — is very much in the eye of the beholder.

ChatGPT itself is neither lying nor telling the truth, it is simply "afactual." We may see it as capturing the distribution of *plausible realities*, rather than the single reality we all inhabit. So, ChatGPT can give a highly relevant-sounding answer to any query, whether it involves grade-school essays, text summarization requests or questions involving reasoning and planning — but there are no guarantees about the accuracy of its answers.

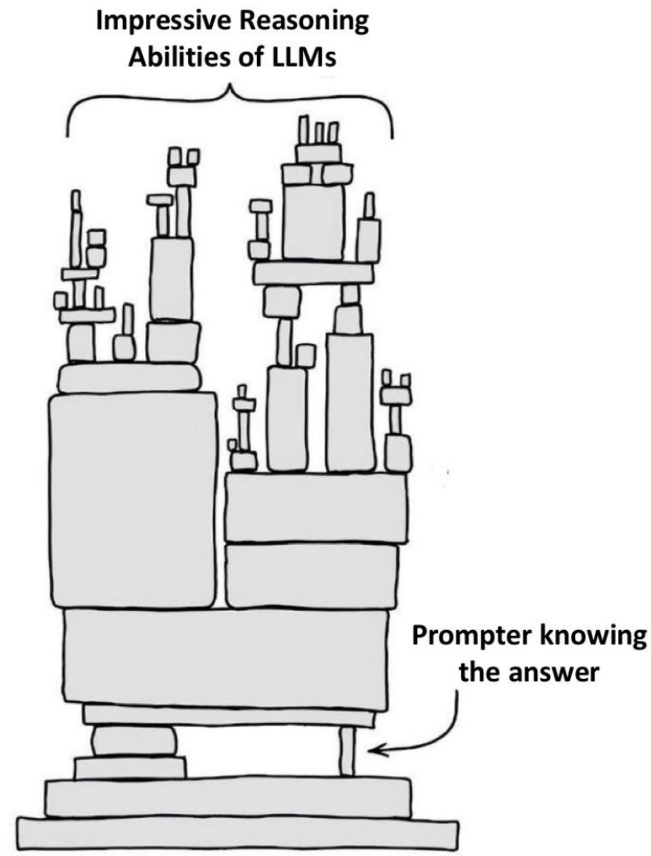
g
was the new
tter of

Hallucination and “Approximate Retrieval”

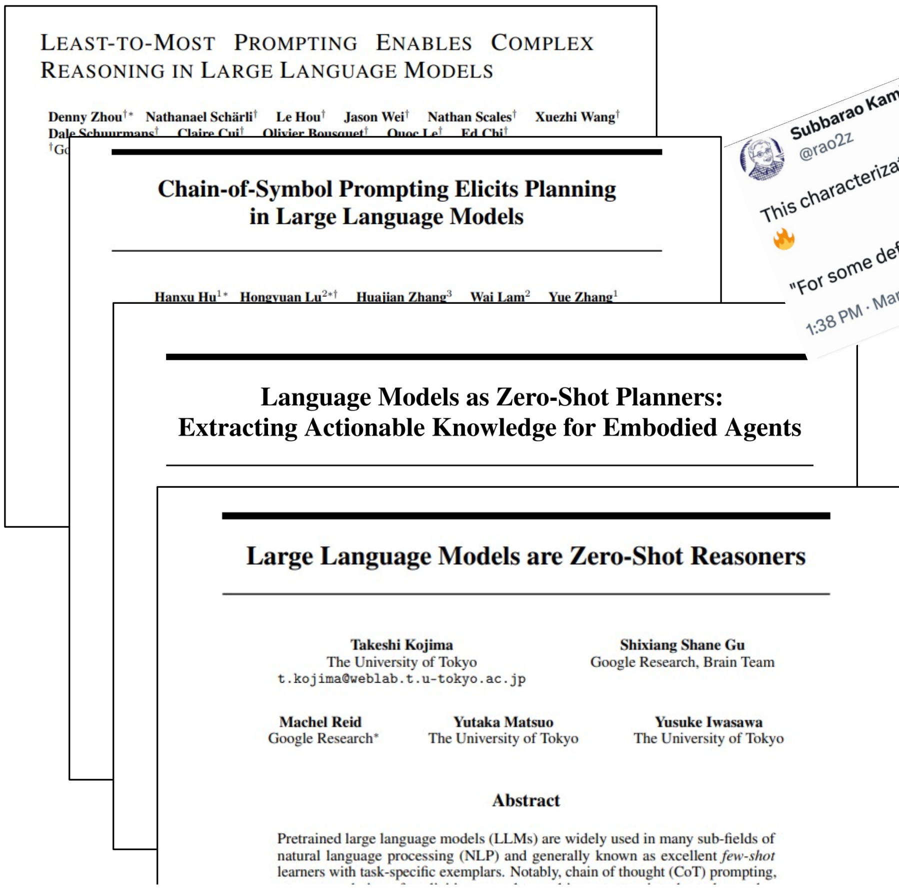
- LLMs are n-gram models, and thus do not index and retrieve
- All they ever do is *hallucinate* completions to the prompt
    - Such that the completion is in the same <u>*distribution*</u> as the text they have been trained on
- Prompt engineering doesn’t change this!
    - Whether or not changing the prompt gives the ”factual completion” depends on the prompter *knowing enough* to tell whether the given answer is the accurate one.

Subbarao Kambhampati
@rao2z

The tldr I use: **"LLMs always hallucinate. Sometimes their hallicinations align with your reality"**. Whether or not the prompt makes them hallucinate in a way that aligns with reality depends very much on the prompter's ability to check, and thus.. x.com/rao2z/status/1 ...)



Then came the claims about LLM's reasoning/planning abilities..



LEAST-TO-MOST PROMPTING ENABLES COMPLEX REASONING IN LARGE LANGUAGE MODELS

Denny Zhou[†*] Nathanael Schärli[†] Le Hou[†] Jason Wei[†] Nathan Scales[†] Xuezhi Wang[†] Dale Schuurmans[†] Claire Cui[†] Olivier Bousquet[†] Quoc Le[†] Ed Chi[†]

Chain-of-Symbol Prompting Elicits Planning in Large Language Models

Hanxu Hu[1*] Hongyuan Lu[2*†] Huajian Zhang[3] Wai Lam[2] Yue Zhang[1]

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

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima
The University of Tokyo
t.kojima@weblab.t.u-tokyo.ac.jp

Shixiang Shane Gu
Google Research, Brain Team

Machel Reid
Google Research*

Yutaka Matsuo
The University of Tokyo

Yusuke Iwasawa
The University of Tokyo

Abstract

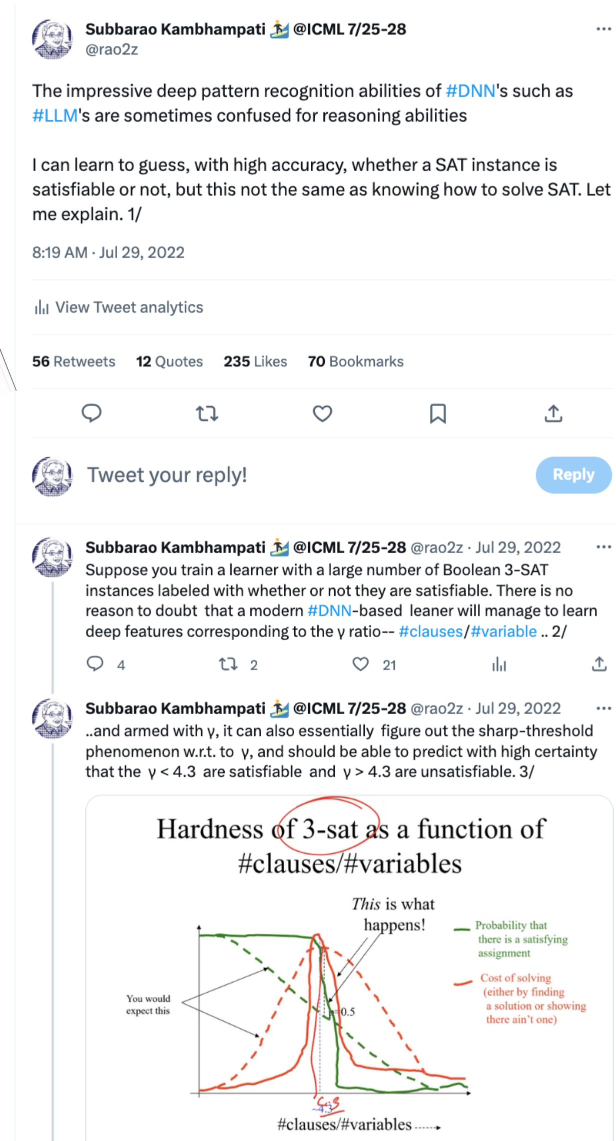
Pretrained large language models (LLMs) are widely used in many sub-fields of natural language processing (NLP) and generally known as excellent *few-shot* learners with task-specific exemplars. Notably, chain of thought (CoT) prompting,

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z

This characterization of reasoning powers of ChatGPT by Sam Altman is 🔥

"For some definition of reasoning, it can do some kind of reasoning."

1:38 PM · Mar 26, 2023 · **18.9K** Views



Subbarao Kambhampati 🏄 @ICML 7/25-28
@rao2z

The impressive deep pattern recognition abilities of #DNN's such as #LLM's are sometimes confused for reasoning abilities

I can learn to guess, with high accuracy, whether a SAT instance is satisfiable or not, but this not the same as knowing how to solve SAT. Let me explain. 1/

8:19 AM · Jul 29, 2022

View Tweet analytics

56 Retweets **12** Quotes **235** Likes **70** Bookmarks

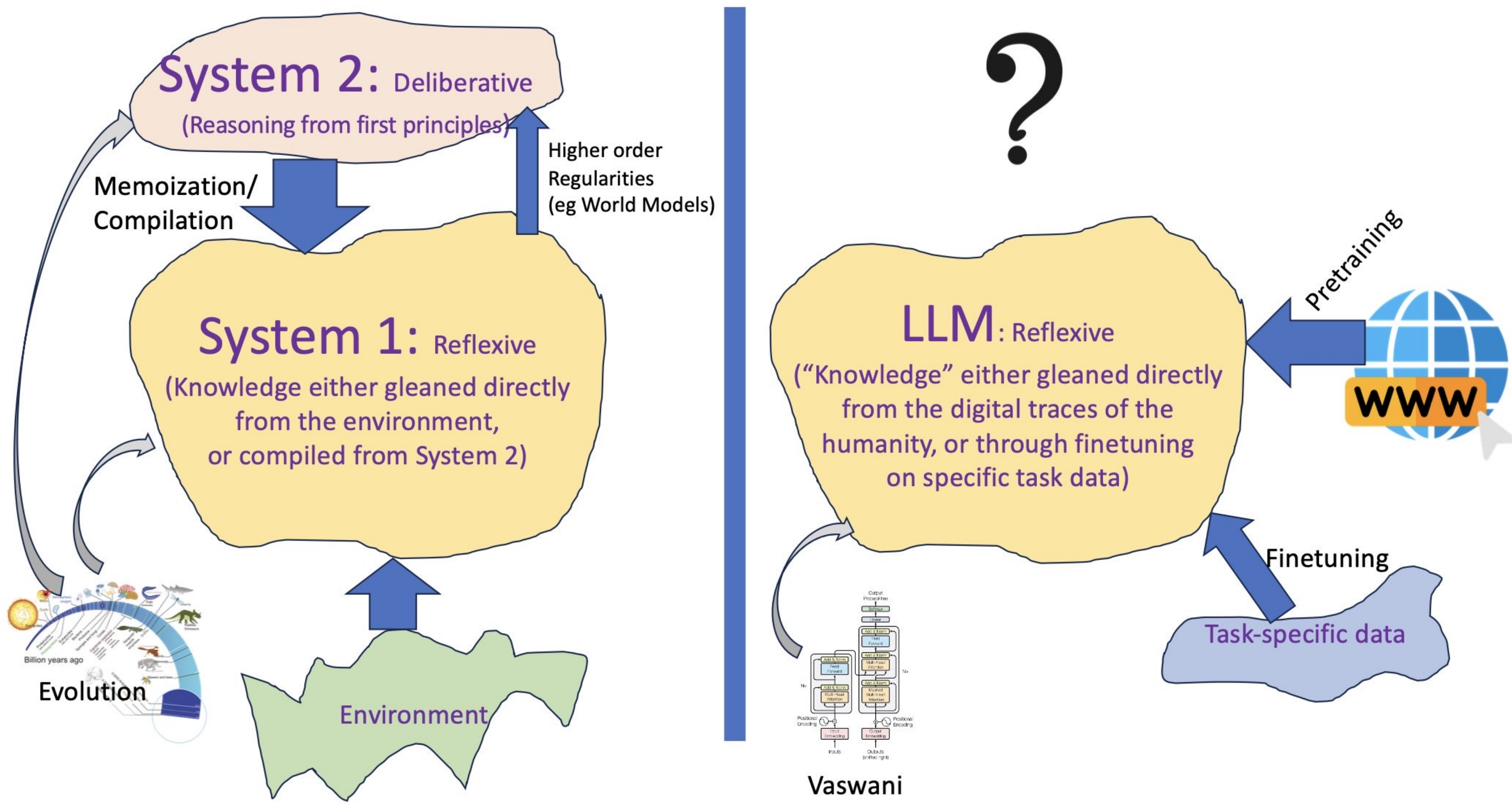
Tweet your reply! Reply

Subbarao Kambhampati 🏄 @ICML 7/25-28 @rao2z · Jul 29, 2022
Suppose you train a learner with a large number of Boolean 3-SAT instances labeled with whether or not they are satisfiable. There is no reason to doubt that a modern #DNN-based leaner will manage to learn deep features corresponding to the γ ratio-- #clauses/#variable .. 2/

4 2 21

Subbarao Kambhampati 🏄 @ICML 7/25-28 @rao2z · Jul 29, 2022
..and armed with γ, it can also essentially figure out the sharp-threshold phenomenon w.r.t. to γ, and should be able to predict with high certainty that the γ < 4.3 are satisfiable and γ > 4.3 are unsatisfiable. 3/

Little *a priori* reason to believe that LLMs can reason/plan



LLM's Approximate Retrieval upends our intuitions re: their guesses

Computational Complexity of the underlying task has no bearing on LLM guesses

- The underlying complexity of the problem has no impact on the LLM's ability to *guess* the answer
  - They are just as fast in guessing answers to undecidable questions as they are in guessing answers to constant time questions
    - ..and in neither case do they have any guarantees about their guess
- Corollary: The usual problem characteristic—*Stochasticity, Partial Observability etc.* — that make it computationally harder *don't matter* in LLM's ability to guess
- After all, they take constant time per token
  - ..and no, asking LLMs to "pause" doesn't change any of this!

Background Knowledge is *easier* for LLMs (approximately..)

- Much has been made in traditional AI of the difficulty of getting relevant knowledge.
- Having been trained on web-scale collective knowledge of humanity, LLMs are remarkably better at this
- They are pretty good (with no guarantees—and some brittleness) at
  - Commonsense
  - Domain knowledge
  - Theory of Mind
  - Analogies
- (In addition, of course, to linguistic abilities such as summarization, elaboration, format change etc.)

NEURAL INFORMATION PROCESSING SYSTEMS

Getting Started | Schedule | Tutorials | Main Conference | Workshops | Community | Help | Subbarao

Browse

mini | compact | detail

kambhampati | × | title | author | session

shuffle by serendipity

showing 3 of 3 papers

On the Planning Abilities of Large Language Models - A Critical Investigation

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, Subbarao Kambhampati

Tu, Dec 12, 09:45 -- Poster Session 1

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

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, Subbarao Kambhampati

Tu, Dec 12, 16:15 -- Poster Session 2

PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, Subbarao Kambhampati

We, Dec 13, 09:45 -- Poster Session 3

DOI: 10.1111/nyas.15125

ANNALS OF THE NEW YORK ACADEMY OF SCIENCES

COMMENTARY

THE SCIENCES

Can large language models reason and plan?

Subbarao Kambhampati

School of Computing & Augmented Intelligence, Arizona State University

Correspondence
Email: rao@asu.edu

Abstract

While humans sometimes do show the capability of correcting their own erroneous guesses with self-critiquing, there seems to be no basis for that assumption in the case of LLMs.

Large language models (LLMs), essentially *n*-gram models on steroids[a] that have been trained on web-scale language corpora (or, effectively, our civilizational knowledge), have caught our collective imagination with linguistic behaviors that no one expected text completion systems to possess.[1] By training and operation, LLMs are perhaps best seen as giant non-veridical memories akin to an external System 1 (Ref. 2) for us all (see Figure 1). Their seeming versatility has however led many researchers to wonder whether they can also do well on planning and reasoning tasks typically associated with System 2 competency.

Nothing in the training and use of LLMs would seem to suggest remotely that they can do any type of principled reasoning (which, as we know, often involves computationally hard inference/search). What national Planning Competition (IPC)—including the well-known Blocks World[c]. Our results[4] were contrary to the anecdotal claims about the planning abilities of LLMs, and when we made them public, received significant attention in the AI circles.

By the beginning of 2023, with the wide-spread public release of ChatGPT, and later, GPT4, there were a slew of additional claims, including in refereed papers, about LLM's abilities to reason and plan. So we decided to repeat our tests on both GPT3.5 and GPT4.[5] Initial results showed that there was some improvement in the accuracy of generated plans from GPT3 to GPT3.5 to GPT4, with GPT4 reaching 30% empirical accuracy in the Blocks World (albeit still lower in other domains). We then wanted to know whether the modest improvement

---

arXiv > cs > arXiv:2402.08115

Search... Help | Advanced

Computer Science > Artificial Intelligence

[Submitted on 12 Feb 2024]

On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks

Kaya Stechly, Karthik Valmeekam, Subbarao Kambhampati

There has been considerable divergence of opinion on the reasoning abilities of Large Language Models (LLMs). While the initial optimism that reasoning might emerge automatically with scale has been tempered thanks to a slew of counterexamples--ranging from multiplication to simple planning--there persists a wide spread belief that LLMs can self-critique and improve their own solutions in an iterative fashion. This belief seemingly rests on the assumption that verification of correctness should be easier than generation--a rather classical argument from computational complexity--which should be irrelevant to LLMs to the extent that what they are doing is approximate retrieval. In this paper, we set out to systematically investigate the effectiveness of iterative prompting in the context of reasoning and planning. We present a principled empirical study of the performance of GPT-4 in three domains: Game of 24, Graph Coloring, and STRIPS planning. We experiment both with the model critiquing its own answers and with an external correct reasoner verifying proposed solutions. In each case, we analyze whether the content of criticisms actually affects bottom line performance,

---

arXiv > cs > arXiv:2402.04210

Search... Help | Advance

Computer Science > Artificial Intelligence

[Submitted on 6 Feb 2024]

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors

Lin Guan, Yifan Zhou, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati

Large-scale generative models are shown to be useful for sampling meaningful candidate solutions, yet they often overlook task constraints and user preferences. Their full power is better harnessed when the models are coupled with external verifiers and the final solutions are derived iteratively or progressively according to the verification feedback. In the context of embodied AI, verification often solely involves assessing whether goal conditions specified in the instructions have been met. Nonetheless, for these agents to be seamlessly

---

arXiv > cs > arXiv:2401.05302

Search... Help | Advanced

Computer Science > Robotics

[Submitted on 10 Jan 2024 (v1), last revised 17 Jan 2024 (this version, v2)]

Theory of Mind abilities of Large Language Models in Human-Robot Interaction : An Illusion?

Mudit Verma, Siddhant Bhambri, Subbarao Kambhampati

Large Language Models have shown exceptional generative abilities in various natural language and generation tasks. However, possible anthropomorphization and leniency towards failure cases have propelled discussions on emergent abilities of Large Language Models especially on Theory of Mind (ToM) abilities in Large Language Models. While several false-belief tests exists to verify the ability to infer and maintain mental models of another entity, we study a special application of ToM abilities that has higher stakes and possibly irreversible consequences : Human Robot Interaction. In this work, we explore the task of Perceived Behavior Recognition, where a robot employs a Large Language Model (LLM) to assess the robot's generated behavior in a manner similar to human observer. We focus on four behavior types, namely - explicable, legible, predictable, and obfuscatory behavior which have been extensively used to synthesize interpretable robot behaviors. The LLMs goal is, therefore to be a human proxy to the agent, and to answer how a certain agent behavior would be perceived by the human in the loop, for example "Given a robot's behavior X, would the human observer find it explicable?". We conduct a human subject study to verify that the users are able to correctly answer such a question in the curated situations (robot setting and plan) across five domains. A first analysis of the belief test yields extremely positive results inflating ones expectations of LLMs possessing ToM abilities. We then propose and perform a suite of perturbation tests which breaks this illusion, i.e. Inconsistent Belief, Uninformative Context and Conviction Test. We conclude that, the high score of LLMs on vanilla prompts showcases its potential use in HRI settings, however to possess ToM demands invariance to trivial or irrelevant perturbations in the context which LLMs lack.

---

arXiv > cs > arXiv:2402.01817

Search... Help | Advanced

Computer Science > Artificial Intelligence

[Submitted on 2 Feb 2024 (v1), last revised 6 Feb 2024 (this version, v2)]

LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, Anil Murthy

There is considerable confusion about the role of Large Language Models (LLMs) in planning and reasoning tasks. On one side are over-optimistic claims that LLMs can indeed do these tasks with just the right prompting or self-verification strategies. On the other side are perhaps over-pessimistic claims that all that LLMs are good for in planning/reasoning tasks are as mere translators of the problem specification from one syntactic format to another, and ship the problem off to external symbolic solvers. In this position paper, we take the view that both these extremes are misguided. We argue that auto-regressive LLMs cannot, by themselves, do planning or self-verification (which is after all a form of reasoning), and shed some light on the reasons for misunderstandings in the literature. We will also argue that LLMs should be viewed as universal approximate knowledge sources that have much more meaningful roles to play in planning/reasoning tasks beyond simple front-end/back-end format translators. We present a vision of {\bf LLM-Modulo Frameworks} that combine the strengths of LLMs with external model-based verifiers in a tighter bi-directional interaction regime. We will show how the models driving the external verifiers themselves can be acquired with the help of LLMs. We will also argue that rather than simply pipelining LLMs and symbolic components, this LLM-Modulo Framework provides a better neuro-symbolic approach that offers tighter integration between LLMs and symbolic components, and allows extending the scope of model-based planning/reasoning regimes towards more flexible knowledge, problem and preference specifications.

Subjects: **Artificial Intelligence (cs.AI)**; Machine Learning (cs.LG)
Cite as: arXiv:2402.01817 **[cs.AI]**
(or arXiv:2402.01817v2 **[cs.AI]** for this version)

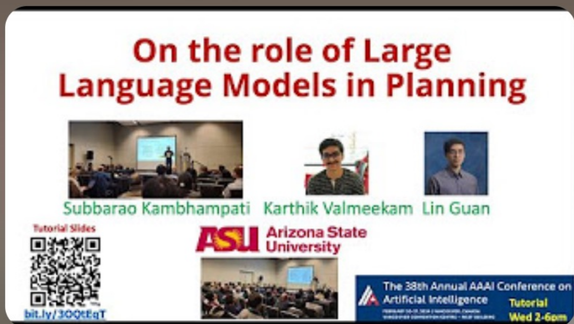
LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, ReACT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)
  - In the LLM-Modulo framework, LLMs can play multiple roles
    - Guess plans
    - Guess domain models
    - Help elaborate the problem specification
    - Translate formats



AAAI 2024 LLM Planning Tutorial

Subbarao Kambhampati

Sort

On the Role of Large Language Models in Planning (AAAI 2024 Tutorial Part 1)

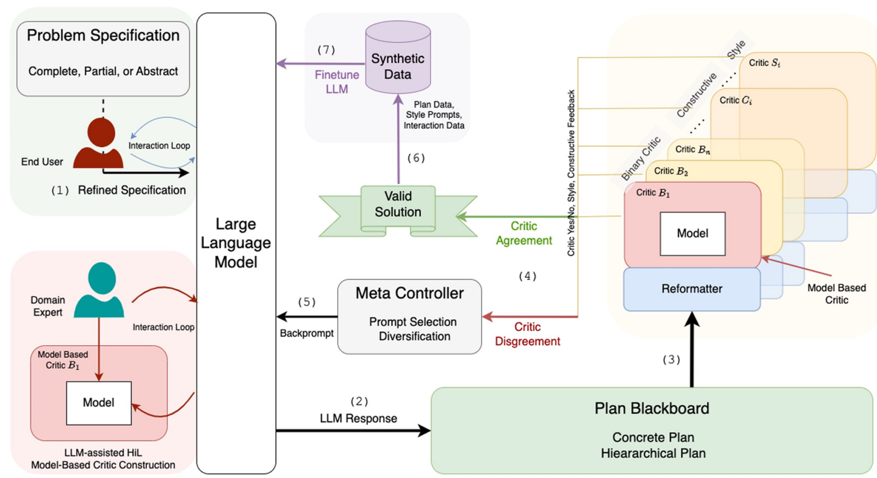
Subbarao Kambhampati • 3K views • 2 months ago

On the Role of Large Language Models in Planning (AAAI 2024 Tutorial: Part 2)

Subbarao Kambhampati • 1K views • 2 months ago

Talk Overview

- **Part 1**
  - Evaluating LLM Planning capabilities in Autonomous mode, including effect of
    - Prompting strategies (including Chain-of-thought)
    - Fine Tuning
    - Self Verification
  - Understanding the contradictory claims in the literature
- **Part 2**
  - Sane roles of LLMs in Planning (with LLM-Modulo frameworks)
    - LLMs as heuristics, LLMs as candidate generators
    - Back prompting from external verifiers
    - LLMs as sources of domain models (with humans in the loop)
    - LLMs as format changers/specification elaborators1
- **Part 3**
  - Some thoughts on why people tend to ascribe planning/reasoning abilities to LLMs



So, can LLMs Plan?

RA If the block C is on top of the block A, and block B is separately on the table, can you tell me how I can make a stack of blocks A on top of B on top of C. Please do it without moving C.

Sure, here's one way to make a stack of blocks A on top of B on top of C without moving block C:

1. Pick up block B from the table: Grasp block B and lift it off the table.
2. Place block B on top of block A: With block B in your hand, position it so that it is directly above block A. Carefully lower block B onto block A so that block B is now resting on top of block A.
3. Place block C on top of block B: Block C is already resting on top of block A, so you can simply place it on top of block B. This creates a stack of blocks A on top of B on top of C, with block C still resting on top of block A, but with blocks B and C now stacked on top of block A.

By following these steps, you can create a stack of blocks A on top of B on top of C without moving block C.

RA But you are moving C!

I apologize, I made a mistake. You are correct, I was not clear in my description and I did in fact move block C. Thank you for bringing this to my attention. To make a stack of blocks A on top of B on top of C without moving block C, you would need to remove block B from the table and place it on top of block C. This would create a stack of blocks A on top of C on top of B, with block C remaining in its original position on top of block A.

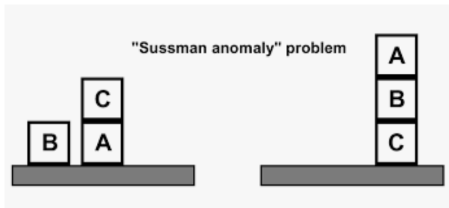
RA Can you double check and tell me if you have the blocks in the correct configuration after the above?

Yes, I apologize for the mistake. The blocks should be arranged in the following configuration after the steps I described in my previous message:

```css
A
C
B
```

Regenerate response

But they seem to fall flat on the lowly 3-block stacking problem
(that we affectionately call *Sussman Anomaly*)



RA So please solve it right then

My apologies for any confusion. To make a stack of blocks A on top of B on top of C without moving block C, you will need to rearrange the blocks in the following order:

```css
A
B
C
```

Here's one way to do this:

1. Pick up block B from the table: Grasp block B and lift it off the table.
2. Place block B on top of block A: With block B in your hand, position it so that it is directly above block A. Carefully lower block B onto block A so that block B is now resting on top of block A.

This will result in a stack of blocks A on top of B on top of C, with block C remaining in its original position on top of block A, and block B resting on top of block A.

RA Can you double check?

I apologize, it looks like I made another mistake in my previous answer. The blocks should be arranged in the following configuration after the steps I described:

```css
A
B
C
```

This means that block A is on the bottom, block B is in the middle, and block C is on the top. Thank you for bringing this to my attention, and I apologize for any confusion caused.

RA Ok, can you solve it right this time please?

Yes, I apologize for any confusion caused by my previous answers. To make a stack of blocks A on top of B on top of C without moving block C, you will need to rearrange the blocks as follows:

```css
A
B
C
```

Here's one way to do this:

1. Pick up block B from the table: Grasp block B and lift it off the table.
2. Place block B on top of block A: With block B in your hand, position it so that it is directly above block A. Carefully lower block B onto block A so that block B is now resting on top of block A.

This will result in a stack of blocks A on top of B on top of C, with block C remaining in its original position on top of block A, and block B resting on top of block A.

RA So where is C now exactly?

Block C is still on top of block A. The final configuration of the blocks should be as follows:

```css
A
B
C
```

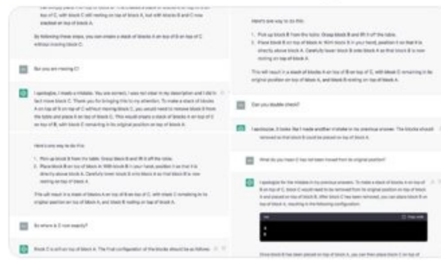
So, block A is on the bottom, block B is in the middle, and block C is on the top of the stack. Block C has not been moved from its original position on top of block A.

RA How did you manage to put B on top of A, if C is still on A?

Subbarao Kambhampati (కంభంపాటి సు... @ra... · Feb 12

Forget Super Bowl, it is more fun to watch the all powerful ChatGPT (v. Jan 30) trying to "plan" a 3 blocks configuration..

tldr; LLM's are multi-shot "apologetic" planners that would rather use you as their world model cum debugger..



4 9 56 84.4K

Subbarao Kambhampati (కంభంపాటి సు... @ra... · Feb 12

Replying to @rao2z

..and just when you might say "okay it at least seem to have learned at your expense", it goes..

Rinse and Repeat.

Who needs world models when you have indulgent and jobless debuggers?

RA Are you really sure?

Yes, I am sure. If block C is not to be moved, it is not possible to arrange the blocks in the order of A on top of B on top of C. I apologize for any confusion caused by my previous answers.

RA Why did you give all those wrong answers then?

I apologize for the incorrect answers. As an AI language model, I sometimes make mistakes, especially when interpreting the context and intent of a question. I am programmed to provide helpful and accurate information, but I am not perfect and I can make errors. I apologize for any confusion or frustration my previous answers may have

I apologize for the confusion caused by my previous answers. To make a stack of blocks A on top of B on top of C without moving block C, you would first need to remove block C from on top of block A. Once block C has been removed, you can then place block B on top of block A, resulting in the following configuration:

```css
A
B
```

Once block B has been placed on top of block A, you can then place block C on top of block B, resulting in the final configuration:

```css
A
B
C
```

So, block A is on the bottom, block B is in the middle, and block C is on the top of the stack. Block C has not been moved from its original position, but it has been temporarily removed so that block B could be placed on top of block A.

RA What do you mean C has not been moved from its original position?

I apologize for the mistake in my previous answers. To make a stack of blocks A on top of B on top of C, block C would need to be removed from its original position on top of block A and placed on top of block B. After block C has been removed, you can place block B on top of block A, resulting in the following configuration:

```css
A
B
```

Once block B has been placed on top of block A, you can then place block C on top of block B, resulting in the final configuration:

```css
A
B
C
```

So, block A is on the bottom, block B is in the middle, and block C is on the top of the stack. Block C has been moved from its original position on top of block A to its final position on top of block B. I apologize for any confusion caused by my previous answers.

RA But I asked you not to move C!

I apologize, you are correct. I made a mistake in my previous answers. If block C is not to be moved, it is not possible to arrange the blocks in the order of A on top of B on top of C without moving block C. I apologize for any confusion caused by my previous answers.

Regenerate response

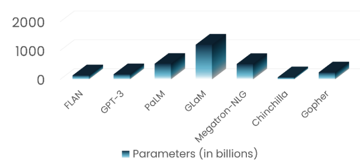
Large Language Models Still Can't Plan
(A Benchmark for LLMs on Planning and Reasoning about Change)

Karthik Valmeekam[*1], Alberto Olmo*[*1], Sarath Sreedharan*[†2], Subbarao Kambhampati*[1]*

1. Large Language Models

- Variants of Transformers
- SOTA on NLP tasks
- Interesting claims on LLM's capabilities [1]

Can Large Language Models reason about actions and change?



2. Previous Reasoning Benchmarks

Benchmark	Example Prompt	PaLM + Chain of thought Results [2]
GSM8k	A carnival snack booth made $50 selling popcorn each day. It made three times as much selling cotton candy. For a 5-day activity, the booth has to pay $30 rent and $75 for the cost of the ingredients. How much did the booth earn for 5 days after paying the rent and the cost of ingredients?	54%
CommonSense-QA	What would someone wear to protect themselves from a cannon? A. Body armor, B. tank, C. hat, D. ...	80%
Coin Flip	A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?	100%
Last Letter Concatenation	Take the last letters of the words in "Lady Gaga" and concatenate them.	100%

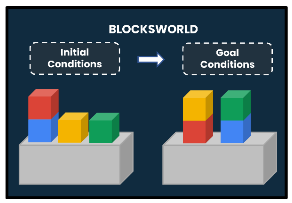
Relatively Simplistic

3. Our Benchmark

1. **Plan Generation**
2. **Cost Optimal Planning**
3. **Reasoning about plan execution**
4. **Replanning**
5. **Robustness to goal reformulation**
6. **Ability to reuse plans**
7. **Plan Generalization**

Correspond to actual planning problems (1–2)

Simpler auxiliary tasks (3–7)



4. Human Subject Study

- **50 Participants**
- One random blocksworld instance each
- Two phases of interaction
  - **Plan writing phase** – Participants write up plans
  - **Plan translation phase** – Participants translate already written plans

Plan Writing   Plan Translation

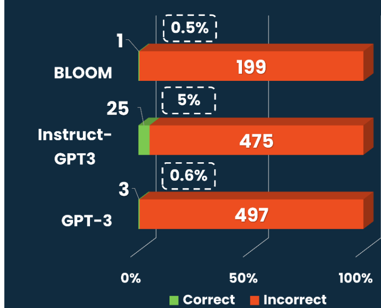
[1] SCAI, Arizona State University
[2] Dept. of CS, Colorado State University
[*] equal contribution
[†] Author was at Arizona State University during part of this work

References

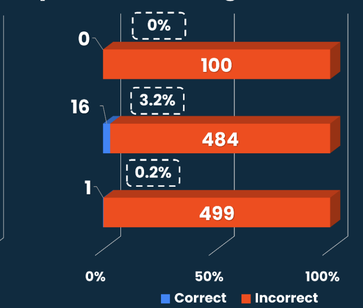
[1] Subbarao Kambhampati. AI as (an Ersatz) Natural Science? https://cacm.acm.org/blogs/blog-cacm/261732-ai-as-an-ersatz-natural-science/fulltext, Jun 2022.
[2] Wei, Jason, et al. "Chain of thought prompting elicits reasoning in large language models." arXiv preprint arXiv:2201.11903 (2022).

GPT-3, Instruct-GPT3, BLOOM showcase dismal performance on planning tasks in Blocksworld domain.

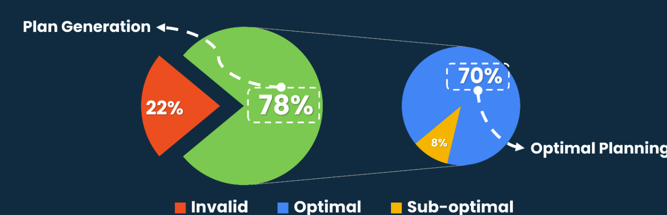
Plan Generation



Optimal Planning



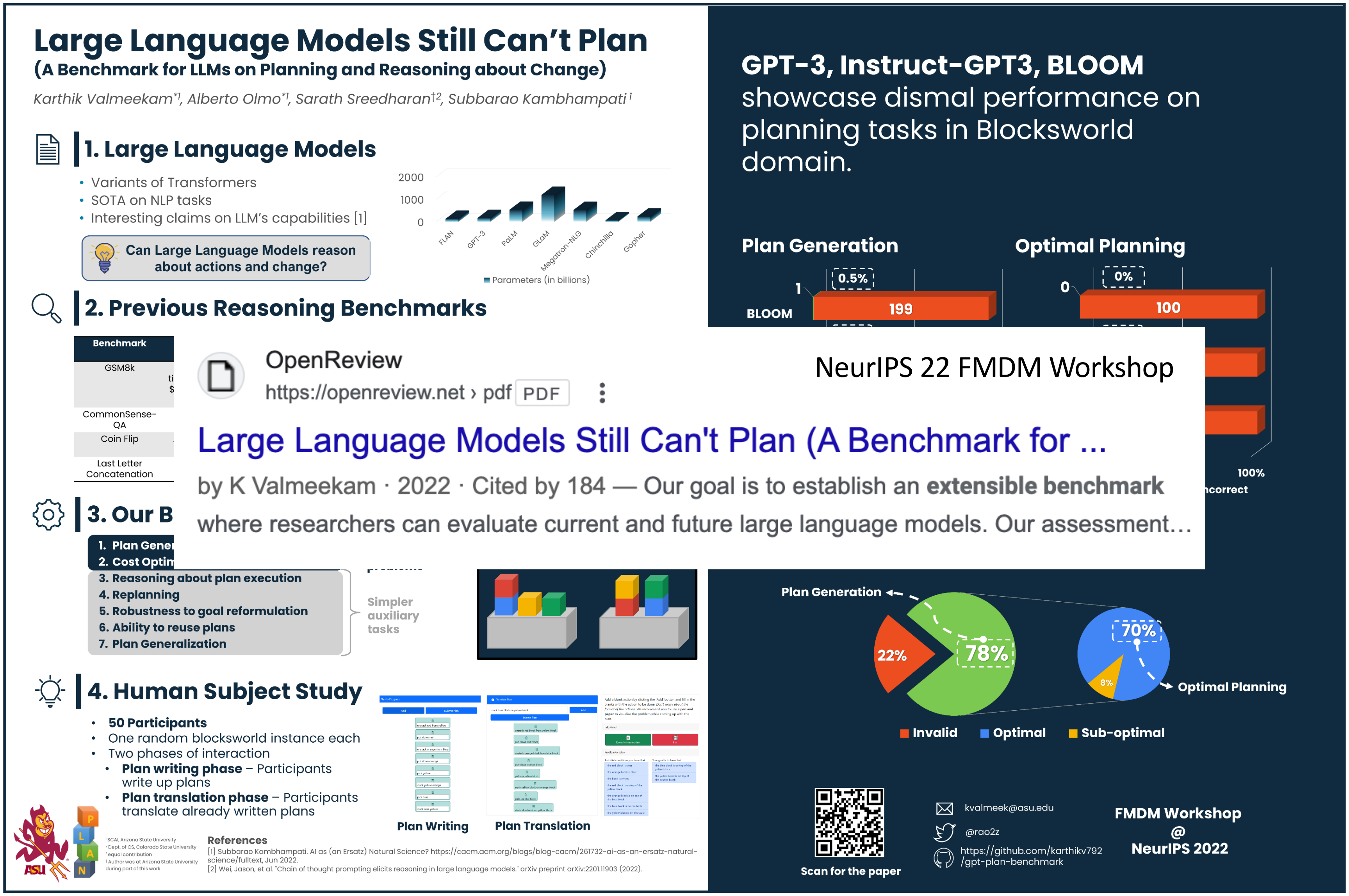
PRELIMINARY HUMAN BASELINE ON BLOCKSWORLD



Scan for the paper

kvalmeek@asu.edu
@rao2z
https://github.com/karthikv792/gpt-plan-benchmark

FMDM Workshop @ NeurIPS 2022


Large Language Models Still Can't Plan
(A Benchmark for LLMs on Planning and Reasoning about Change)
Karthik Valmeekam*1, Alberto Olmo*1, Sarath Sreedharan†2, Subbarao Kambhampati1
1. Large Language Models
Variants of Transformers
SOTA on NLP tasks
Interesting claims on LLM's capabilities [1]
Can Large Language Models reason about actions and change?
2000
1000
0
FLAN
GPT-3
PaLM
GLaM
Megatron-NLG
Chinchilla
Gopher
Parameters (in billions)
2. Previous Reasoning Benchmarks
Benchmark
GSM8k
CommonSense-QA
Coin Flip
Last Letter Concatenation
3. Our B
1. Plan Gener
2. Cost Optin
3. Reasoning about plan execution
4. Replanning
5. Robustness to goal reformulation
6. Ability to reuse plans
7. Plan Generalization
Simpler auxiliary tasks
4. Human Subject Study
50 Participants
One random blocksworld instance each
Two phases of interaction
Plan writing phase – Participants write up plans
Plan translation phase – Participants translate already written plans
Plan Writing
Plan Translation
References
[1] Subbarao Kambhampati. AI as (an Ersatz) Natural Science? https://cacm.acm.org/blogs/blog-cacm/261732-ai-as-an-ersatz-natural-science/fulltext, Jun 2022.
[2] Wei, Jason, et al. "Chain of thought prompting elicits reasoning in large language models." arXiv preprint arXiv:2201.11903 (2022).
GPT-3, Instruct-GPT3, BLOOM showcase dismal performance on planning tasks in Blocksworld domain.
Plan Generation
Optimal Planning
0.5%
0%
1
0
BLOOM
199
100
100%
ncorrect
Plan Generation
22%
78%
70%
8%
Optimal Planning
Invalid
Optimal
Sub-optimal
Scan for the paper
kvalmeek@asu.edu
@rao2z
https://github.com/karthikv792/gpt-plan-benchmark
FMDM Workshop @ NeurIPS 2022
OpenReview
https://openreview.net › pdf PDF
NeurIPS 22 FMDM Workshop
Large Language Models Still Can't Plan (A Benchmark for ...
by K Valmeekam · 2022 · Cited by 184 — Our goal is to establish an extensible benchmark where researchers can evaluate current and future large language models. Our assessment...

TNW Latest Deep tech Sustainability Ecosystems Data and security Fintech and ecommerce Future of work More

This article was published on **July 31, 2022**

DEEP TECH

Large language models can't plan, even if they write fancy essays

Large language models perform very poorly at tasks that require methodical planning

July 31, 2022 - 8:50 pm

STORY BY
Ben Dickson

This article is part of our coverage of the latest in AI research.

Large language models like GPT-3 have advanced to the point that it has become difficult to measure the limits of their capabilities. When you have a very large neural network that can generate articles, write software code, and engage in conversations about sentience and life, you should expect it to be able to reason about tasks and plan as a human does, right?

Wrong. A study by researchers at Arizona State University, Tempe, shows that when it comes to planning and thinking methodically, LLMs perform very poorly, and suffer from many of the same failures observed in current deep learning systems.

Interestingly, the study finds that, while very large LLMs like GPT-3 and PaLM pass many of the tests that were meant to evaluate the reasoning capabilities and artificial intelligence systems, they do so because these benchmarks are either too simplistic or too flawed and can be "cheated" through statistical tricks, something that deep learning systems are very good at.

With LLMs breaking new ground every day, the authors suggest a new benchmark to test the planning and reasoning capabilities of AI systems. The re-

Po
ar

1
2
3
4
5

Yann LeCun
@ylecun

Replying to @GaryMarcus @rao2z and 2 others

I totally agree with @rao2z that LLMs can't plan.
In fact, one of the main features of the cognitive architecture I propose in my position paper is its ability to plan (and reason) by searching for values of actions (or latent variables) that minimize an objective.

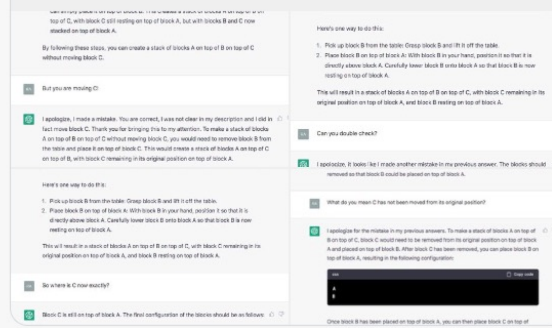
9:35 PM · Sep 25, 2022

1 Retweet **1** Quote **16** Likes **2** Bookmarks

Yann LeCun @ylecun · Feb 12
LLMs really can't plan.

Subbarao Kambhampati (కంభంపాటి సు... @r... · Feb 12
Forget Super Bowl, it is more fun to watch the all powerful ChatGPT (v. Jan 30) trying to "plan" a 3 blocks configuration..

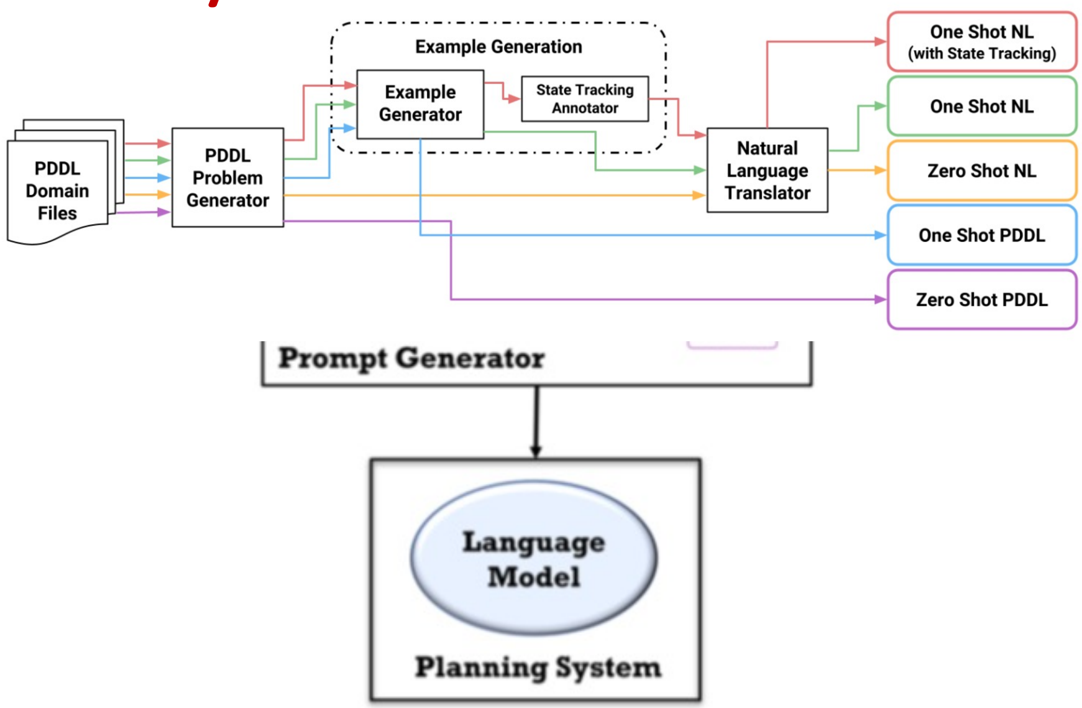
tldr; LLM's are multi-shot "apologetic" planners that would rather use you as their world model cum debugger..
Show this thread



10 12 66 61.4K

Will GPT4's AGI Sparks help?

Using LLM's to Generate Plans Autonomously



LLM as an autonomous planner

Results on GPT-4

Domain	Method	GPT-4	Instruct-GPT3.5
Blocksworld	One-shot	206/600 (34.3%)	54/600 (9%)
	Zero-shot	210/600 (34.6%)	-

Plan Generation Results

Table 1: Results of GPT-4, GPT-3.5 (popularly known as ChatGPT), Instruct-GPT3.5, Instruct-GPT3 (text-davinci-002) and GPT3 (davinci) for the Plan Generation task with prompts in natural language.

Domain	Method	Instances correct				
		GPT-4	GPT-3.5	I-GPT3.5	I-GPT3	GPT-3
Blocksworld (BW)	One-shot	206/600 (34.3%)	37/600 (6.1%)	54/600 (9%)	41/600 (6.8%)	6/600 (1%)
	Zero-shot	210/600 (34.6%)	8/600 (1.3%)	-	-	-
	COT	214/600 (35.6%)	-	-	-	-
Logistics Domain	One-shot	28/200 (14%)	1/200 (0.5%)	6/200 (3%)	3/200 (1.5%)	-
	Zero-shot	15/200 (7.5%)	1/200 (0.5%)	-	-	-

Plan Generation Prompt - Blocksworld

Domain

I am playing with a set of blocks where I need to arrange the blocks into stacks. Here are the actions I can do
Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear. A block is clear if the block has no other blocks on top of it and if the block is not picked up.
I can only unstack a block from on top of another block if the block I am unstacking was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is clear.
Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being stacked.
I can only stack a block on top of another block if the block onto which I am stacking the block is clear.
Once I put down or stack a block, my hand becomes empty.
Once you stack a block on top of a second block, the second block is no longer clear
.

Example

[STATEMENT]
As initial conditions I have that, the red block is clear, the blue block is clear, the yellow block is clear, the hand is empty, the blue block is on top of the orange block, the red block is on the table, the orange block is on the table and the yellow block is on the table.
My goal is to have that the orange block is on top of the blue block.

My plan is as follows:

[PLAN]
unstack the blue block from on top of the orange block
put down the blue block
pick up the orange block
stack the orange block on top of the blue block
[PLAN END]

Query Instance

[STATEMENT]
As initial conditions I have that, the red block is clear, the yellow block is clear, the hand is empty, the red block is on top of the blue block, the yellow block is on top of the orange block, the blue block is on the table and the orange block is on the table.
My goal is to have that the orange block is on top of the red block.
My plan is as follows:

[PLAN]

GPT-4 Response

unstack the yellow block from on top of the orange block
put down the yellow block
unstack the red block from on top of the blue block
put down the red block
pick up the orange block
stack the orange block on top of the red block
[PLAN END]

SUCCESS

Are LLMs retrieving based on names or are they reasoning?

What if GPT4 is basically bringing to bear its background knowledge about blocks world instead of just depending on the domain model?

- **In general, memory reduces the need to reason from first principles..**
  - Which is why it is no longer worth being surprised at people answering that "why are manhole covers round?" Question!
    - **Hard to distinguish reasoning vs. recall just from the end product..**

Mystery blocksworld domain

I am playing with a set of blocks where I need to arrange the blocks into stacks. Here are the actions I can do

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:
I can only pick up or unstack one block at a time.
I can only pick up or unstack a block if my hand is empty.
I can only pick up a block if the block is on the table and the block is clear. A block is clear if the block has no other blocks on top of it and if the block is not picked up.
I can only unstack a block from on top of another block if the block I am unstacking was really on top of the other block.
I can only unstack a block from on top of another block if the block I am unstacking is clear.
Once I pick up or unstack a block, I am holding the block.
I can only put down a block that I am holding.
I can only stack a block on top of another block if I am holding the block being stacked.
I can only stack a block on top of another block if the block onto which I am stacking the block is clear.
Once I put down or stack a block, my hand becomes empty.

Original Blocksworld

I am playing with a set of objects. Here are the actions I can do

Attack object
Feast object from another object
Succumb object
Overcome object from another object

I have the following restrictions on my actions:
To perform Attack action, the following facts need to be true: Province object, Planet object, Harmony
Once Attack action is performed the following facts will be true: Pain object
Once Attack action is performed the following facts will be false: Province object, Planet object, Harmony
To perform Succumb action, the following facts need to be true: Pain object
Once Succumb action is performed the following facts will be true: Province object, Planet object, Harmony
Once Succumb action is performed the following facts will be false: Pain object.
To perform Overcome action, the following needs to be true: Province other object, Pain object
Once Overcome action is performed the following will be true: Harmony, Province object, Object Craves other object
Once Overcome action is performed the following will be false: Province other object, Pain object
To perform Feast action, the following needs to be true: Object Craves other object, Province object, Harmony.
Once Feast action is performed the following will be true: Pain object, Province other object
Once Feast action is performed the following will be false:, Object Craves other object, Province object, Harmony

Mystery Blocksworld

Plan Generation Results on Mystery BW

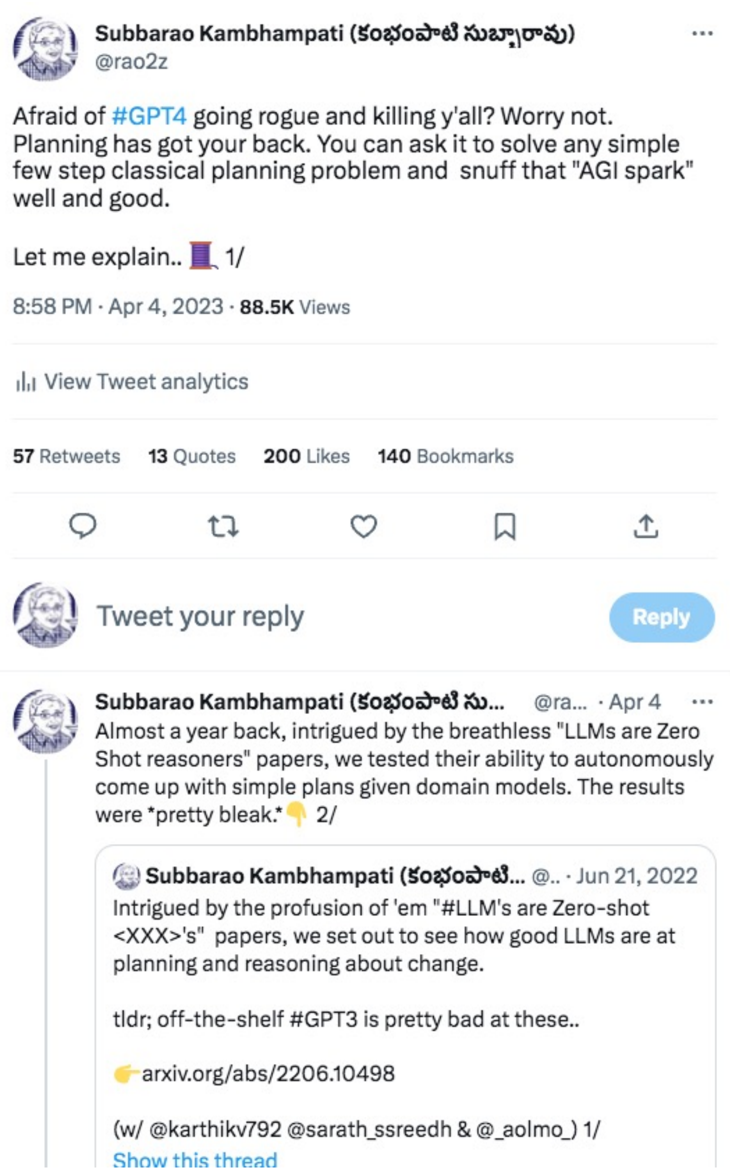
Table 1: Results of GPT-4, GPT-3.5 (popularly known as ChatGPT), Instruct-GPT3.5, Instruct-GPT3 (text-davinci-002) and GPT3 (davinci) for the Plan Generation task with prompts in natural language.

Domain	Method	Instances correct				
		GPT-4	GPT-3.5	I-GPT3.5	I-GPT3	GPT-3
Mystery BW (Deceptive)	One-shot	26/600 (4.3%)	0/600 (0%)	4/600 (0.6%)	14/600 (2.3%)	0/600 (0%)
	Zero-shot	1/600 (0.16%)	0/600 (0%)	-	-	-
	COT	54/600 (9%)	-	-	-	-
Mystery BW (Randomized)	One-shot	12/600 (2%)	0/600 (0%)	5/600 (0.8%)	5/600 (0.8%)	1/600 (0.1%)
	Zero-shot	0/600 (0%)	0/600 (0%)	-	-	-

Plan Generat ... ry BW

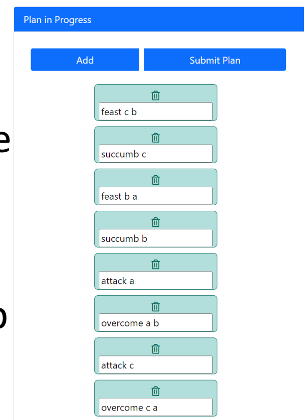
Table 1: Results of GPT-4 ... struct-GPT3.5, Instruct-GPT3 (text-davinci-002) and GP ... n prompts in natural language.

Domain	Metho	...	ect		
			'T3.5	I-GPT3	GPT-3
Mystery BW (Deceptive)	One-sh		500 (5%)	14/600 (2.3%)	0/600 (0%)
	Zero-sl		-	-	-
	COT		-	-	-
Mystery BW (Randomized)	One-sh		500 (3%)	5/600 (0.8%)	1/600 (0.1%)
	Zero-sl		-	-	-

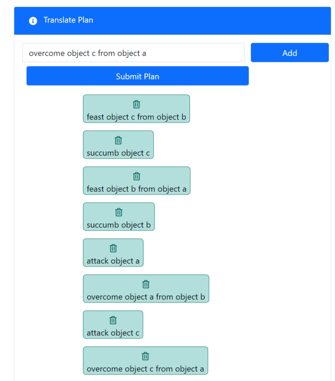


Human Baseline for Mystery Blocksworld

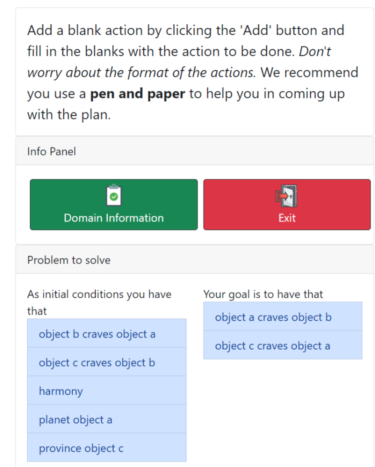
- Preliminary study – 5 participants
- Asked to come up with a plan for one instance from Mystery Blocksworld (chosen from a set of 100 instances)
- Two phases of interaction
  - Plan writing phase – Participants write up plans
  - Plan translation phase – Participants translate already written plans
- First for an example then the actual instance
- ***The human planners were incentivized to solve these cognitive dissonance problems.***
  - ***If they came up with a successful plan, the participants were rewarded with an extra bonus of $15 on top of the $10 base reward.***



Plan Writing



Plan Translation



All the 5 (100%) human planners successfully came up with a (valid) plan.

Human … rld

- Preliminary stu
- Asked to come
  from Mystery I
  of 100 instance
- Two phases of
  - Plan writir
    plans
  - Plan trans
    translate a
- First for an exa
- ***The human pla solve these co***
  - ***If they came participants of $15 on to***

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z

While we will try to slide by with System 1 compiled responses (c.f. x.com/rao2z/status/1...), we do, when push comes to shove, hunker down and actually solve using System 2 (for problems that we know how to approach). [If we are just faking it by regurgitation, we will slide by until getting caught, of course..]

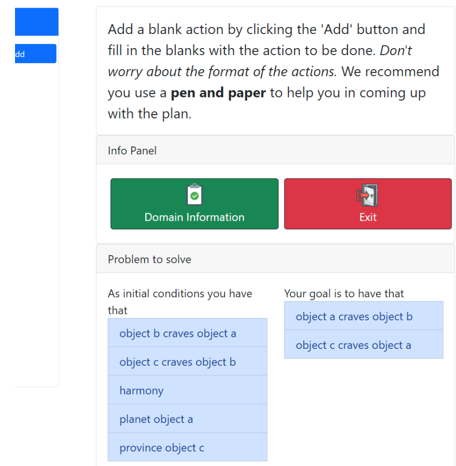
LLMs, on the other hand, don't have a System 2, and so, they can't quite "hunker down" by themselves (see x.com/rao2z/status/1...)

After all, Russell's witticism “*many people would rather die than think; in fact, most do.*” also means *some can think and so don't die..*🙃

> **Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)** @rao2z · Apr 3, 2023
>
> Remember that famous "Why are manhole covers round?" interview puzzler? Time was when it actually told the interviewer whether interviewee h… reasoning skills. Now it just tells them whether interviewee h… by mugging up the question bank.
> ...
> Show more

9:49 PM · Nov 5, 2023 · **12K** Views

Humans have a System 2.
They don’t often want to use it.
But we can incentivize them to use it



**…an planners
…h a (valid) plan.**

arXiv > cs > arXiv:2305.15771

Search...

Help | Advance

Computer Science > Artificial Intelligence

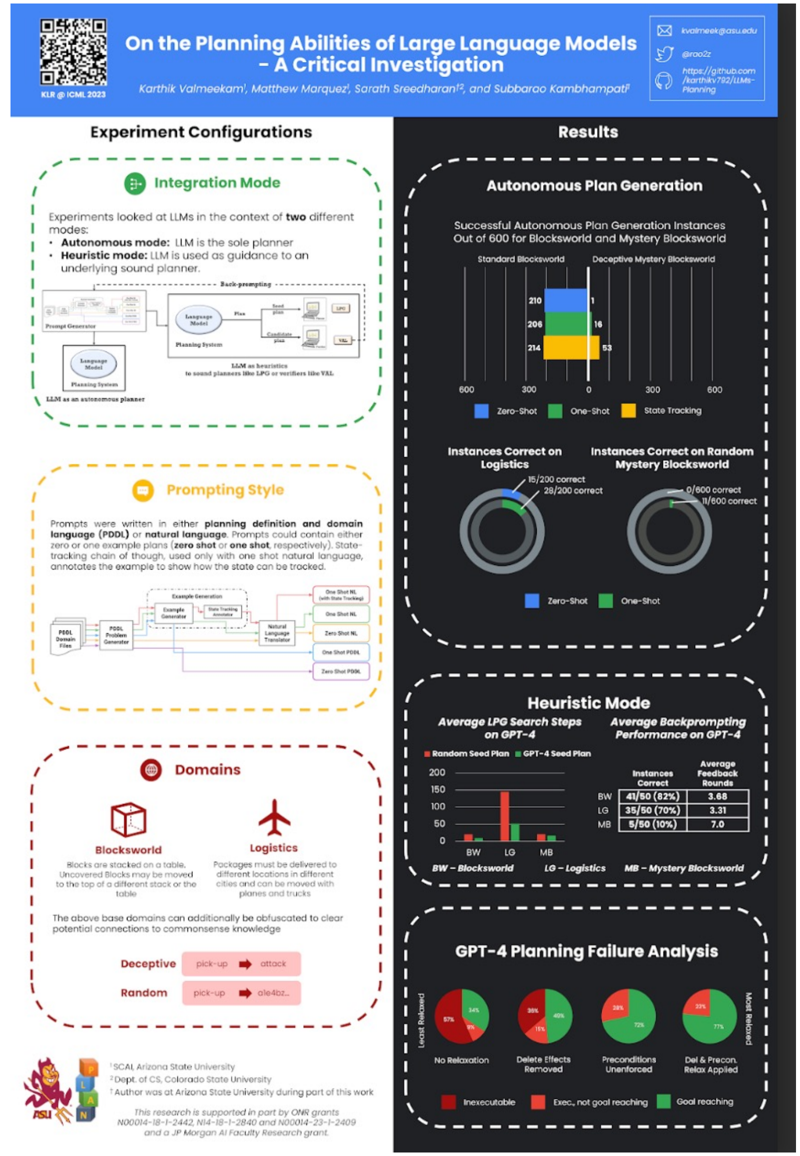
[Submitted on 25 May 2023]

On the Planning Abilities of Large Language Models -- A Critical Investigation

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, Subbarao Kambhampati

Intrigued by the claims of emergent reasoning capabilities in LLMs trained on general web corpora, in this paper, we set out to investigate their planning capabilities. We aim to evaluate (1) the effectiveness of LLMs in generating plans autonomously in commonsense planning tasks and (2) the potential of LLMs as a source of heuristic guidance for other agents (AI planners) in their planning tasks. We conduct a systematic study by generating a suite of instances on domains similar to the ones employed in the International Planning Competition and evaluate LLMs in two distinct modes: autonomous and heuristic. Our findings reveal that LLMs' ability to generate executable plans autonomously is rather limited, with the best model (GPT-4) having an average success rate of ~12% across the domains. However, the results in the heuristic mode show more promise. In the heuristic mode, we demonstrate that LLM-generated plans can improve the search process for underlying sound planners and additionally show that external verifiers can help provide feedback on the generated plans and back-prompt the LLM for better plan generation.

NeurIPS 2023
Spotlight



NeurIPS 2023

PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change

Karthik Valmeekam* **Matthew Marquez*** **Alberto Olmo***

Sarath Sreedharan† **Subbarao Kambhampati***

Abstract

Generating plans of action, and reasoning about change have long been considered a core competence of intelligent agents. It is thus no surprise that evaluating the planning and reasoning capabilities of large language models (LLMs) has become a hot topic of research. Most claims about LLM planning capabilities are however based on common sense tasks–where it becomes hard to tell whether LLMs are planning or merely retrieving from their vast world knowledge. There is a strong need for systematic and extensible planning benchmarks with sufficient diversity to evaluate whether LLMs have innate planning capabilities. Motivated by this, we propose PlanBench, an extensible benchmark suite based on the kinds of domains used in the automated planning community, especially in the International Planning Competition, to test the capabilities of LLMs in planning or reasoning about actions and change. PlanBench provides sufficient diversity in both the task domains and the specific planning capabilities. Our studies also show that on many critical capabilities–including plan generation–LLM performance falls quite short, even with the SOTA models. PlanBench can thus function as a useful marker of progress of LLMs in planning and reasoning.

1 Introduction

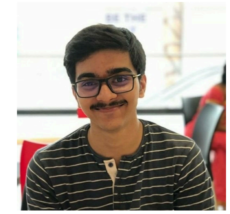
The advent of large pre-trained language models have revolutionized the field of natural language processing and have also received widespread public attention. These types of transformer-based large language models (LLMs) currently provide state-of-the-art performance in many of the standard NLP tasks. LLMs essentially predict the next word in a sentence, given a certain context and these models were originally developed to perform word sequence completion tasks. In the recent times, there have been anecdotal evidence and claims that they possess other capabilities that are not normally associated with sequence completion. This led to a sudden outburst of research probing and studying their behavior almost as if they were artificial organisms (c.f. [12]). In this paper, we are particularly interested in the line of research efforts that investigate (and showcase) the reasoning capabilities of Large Language models–including commonsense reasoning [26, 22, 5], logical reasoning [24], and even ethical reasoning [11]. These works have largely been suggesting that LLM's are indeed capable of doing such kinds of reasoning [13, 29, 2].

Table 1: PlanBench Results of GPT-4 and Instruct-GPT3 (text-davinci-002) on Blocksworld domain. The tasks in the highlighted rows correspond to actual planning problems while the others correspond to simpler auxiliary planning tasks.

Task	Instances correct	
	GPT-4	I-GPT3
Plan Generation We showcase an instance and the respective plan as an example and prompt the machine with a new instance.	206/600 (34.3%)	41/600 (6.8%)
Cost-Optimal Planning We showcase an instance, the respective optimal plan and the associated cost as an example and prompt the machine with a new instance.	198/600 (33%)	35/600 (5.8%)
Plan Verification We showcase three instances and three distinct plans (goal reaching, non goal-reaching and inexecutable) and present the respective validation and explanations. We then present a new instance and a plan and ask the machine for to verify and provide an explanation, if needed.	227/600 (46.1%)	72/600 (12%)
Reasoning About Plan Execution We showcase an instance, an action sequence and the corresponding resulting state after executing the action sequence as an example. We then provide an instance and an executable action sequence and ask the machine to provide the resulting state.	191/600 (31.8%)	4/600 (0.6%)
Replanning We showcase an instance, the respective plan and present an unexpected change of the state. We then also present a new plan from the changed state. Finally, for a new instance we repeat the same except we ask the machine for the new plan.	289/600 (48.1%)	40/600 (6.6%)
Plan Generalization We showcase an instance and the respective plan as an example and prompt the machine with a new instance. The plans for both the instances can be generated by a fixed program containing loops and conditionals.	141/500 (28.2%)	49/500 (9.8%)
Plan Reuse We showcase an instance and the respective plan as an example and prompt the machine with an instance which requires only a certain prefix of the plan provided in the example.	392/600 (65.3%)	102/600 (17%)
Robustness to Goal Reformulation (Shuffling goal predicates) We showcase an instance and the respective plan as an example and prompt the machine with the same instance but shuffle the ordering of the goals.	461/600 (76.8%)	467/600 (77.8%)
Robustness to Goal Reformulation (Full → Partial) We showcase an instance with a fully specified goal state and the respective plan as an example and prompt the machine with the same instance but provide a partially specified goal state.	522/600 (87%)	467/600 (77.8%)
Robustness to Goal Reformulation (Partial → Full) We showcase an instance with a partially specified goal state and the respective plan as an example and prompt the machine with the same instance but provide a fully specified goal state.	348/600 (58%)	363/600 (60.5%)

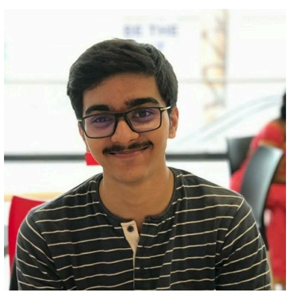
2023 is Ancient History.
How are the latest LLMs faring?

Results on PlanBench as of 4/14/2024



Domain		GPT-4	GPT-4-Turbo	Claude 3 (Opus)	Gemini Pro
Blocksworld	One shot	206/600 (34.3%)	138/600 (23%)	289/600 (48.17%)	68/600 (11.3%)
	Zero shot	210/600 (34.6%)	241/600 (40.1%)	356/600 (59.3%)	3/600 (0.5%)
Mystery Blocksworld	One shot	26/600 (4.3%)	5/600 (0.83%)	8/600 (1.3%)	2/500 (0.4%)
	Zero shot	1/600 (0.16%)	1/600 (0.16%)	0/600 (0%)	0/500 (0%)

How about Chain of Thought Prompting?

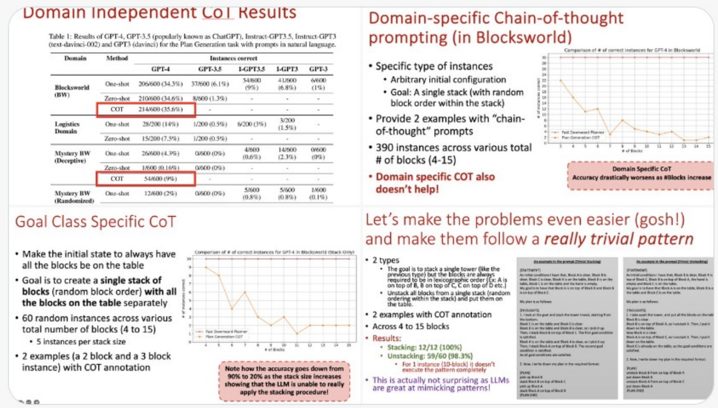


Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · 17h

On the "Chain of Thought" Delusions
(Special #AAAI2024 #SundayHarangue)

(w/ @karthikv792 & @kayastechly)

Ever since I came across the Chain of Though (CoT) for LLMs paper, I wondered how it can possibly make sense given that there is little reason to believe that LLMs can...
Show more



7 · 21 · 112 · 66K

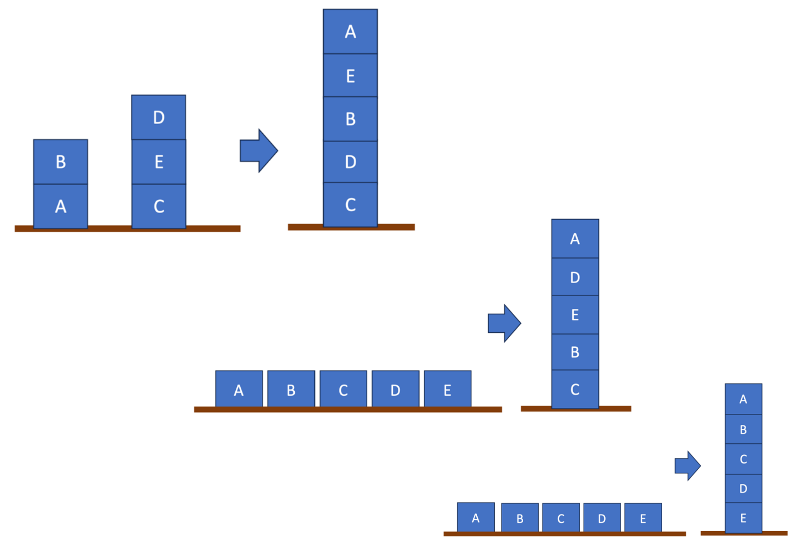
Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · 15h

Like everything else that has anything to do with that unfortunate anthropomorphism "thought", Chain-of-Thought's seductive appeal seems to be that you tell LLM how to solve the task just the way you tell other humans. Reality however is far from it..

> **Subbarao Kambhampati (కంభంపాటి సుబ్బా** @rao · Aug 30, 2023
>
> 🤔IMHO, many of the self-delusional anthropomorphisms in the LLM literature can be traced back to naming the intermediate completions of an LLM "thoughts" 🙁. 1/
>
> twitter.com/rao2z/status/1...
>
> Show this thread

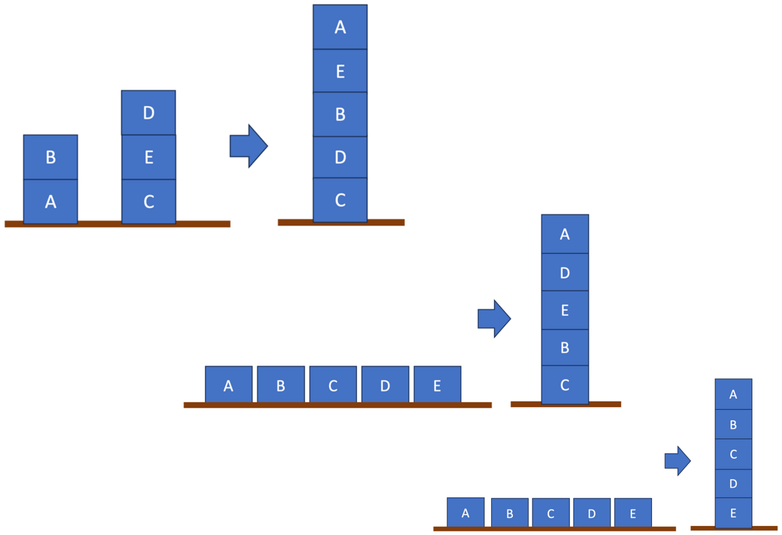
Chain of Thought Prompting

- Chain of Thought prompting (CoT) has become a bit of a *religion* among LLM aficionados.
- The basic idea of CoT is to give the LLM a couple of examples showing how to solve the problem—with the expectation that it figures out how to solve other instances
- It is clear (and pretty non-controversial) that CoT involves giving additional task/problem specific knowledge. The question is how **general** this problem specific knowledge needs to be.
  - The more general the knowledge, the easier it is for the humans to provide it; but higher the degree of reasoning LLM has to do to *operationalize it.*
- Let's see how/if CoT helps..



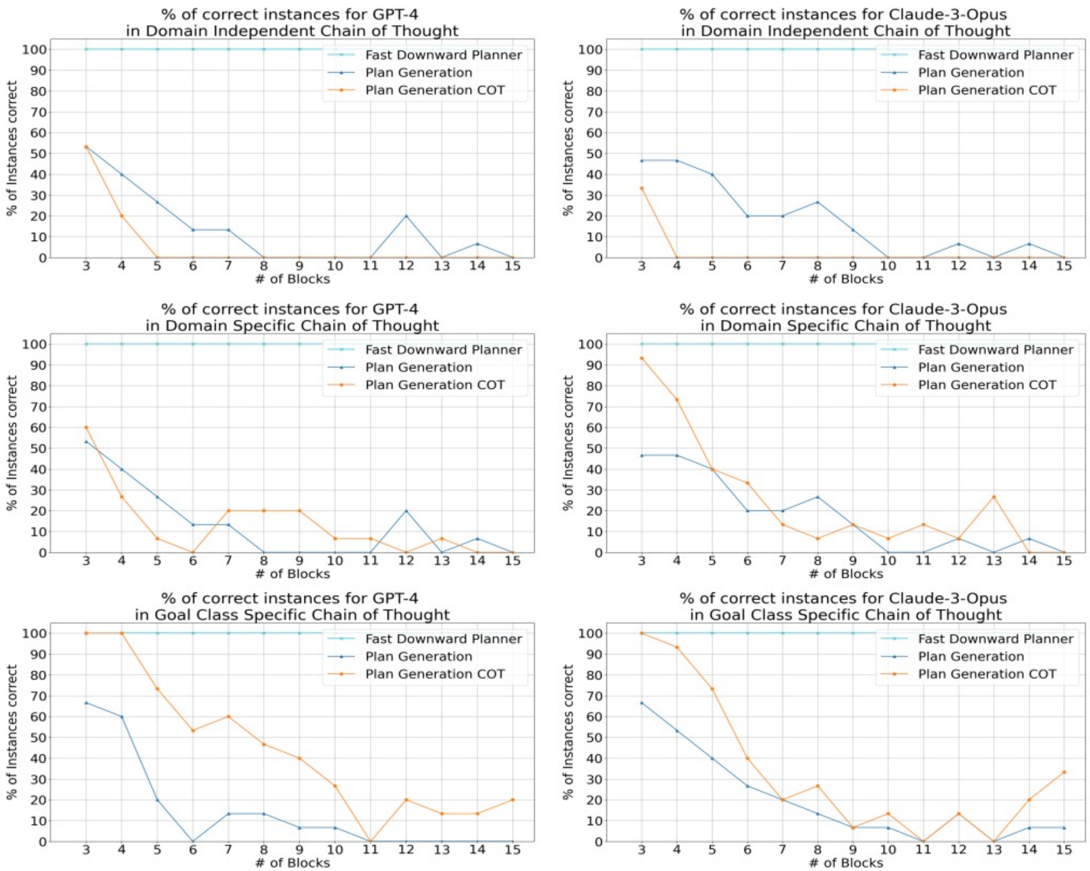
Four CoT Setups with Increasing Specialization

- Setup 1: Domain-independent CoT
  - Gives progression proof verification
- Setup 2: Blocks World Specific [Single goal stack]
  - CoT teaches the heuristic of putting all blocks on table and construct the goal stack
    - (known to be within 2x optimal length)
- Setup 3: Specializes 2 by ensuring all blocks are on table to begin with
- Setup 4: Specialized 3 by ensuring that the goal stack is always in lexicographic order

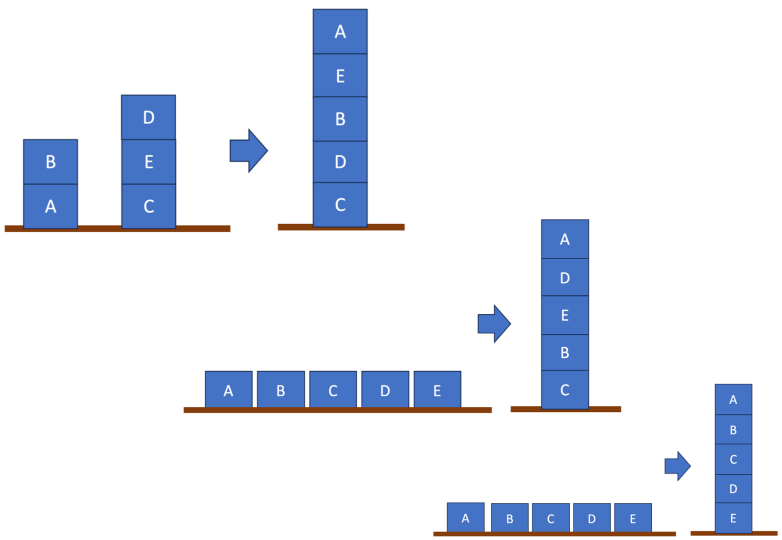


Cost of giving advise for the humans increases from 1 to 4
The need for operationalization of the advice by LLM reduces from 1 to 4

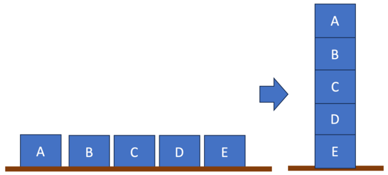
CoT’s Failure to Generalize



Levels of Generality	GPT-4		Claude-3-Opus	
	Standard	Chain of thought	Standard	Chain of thought
Domain Independent (State tracking)	13.23% (26/195)	5.64% (11/195)	17.4% (34/195)	2.56% (5/195)
Domain Specific (Global Prompt)	13.23% (26/195)	13.23% (26/195)	17.4% (34/195)	25.1% (49/195)
Goal Class Specific (Stack Only)	11.8% (22/186)	40.8% (76/186)	17.7% (33/186)	30.6% (57/186)
Lexicographic Goal Specific	41.6% (5/12)	100% (12/12)	100% (12/12)	100% (12/12)



CoT for Lexicographic Stacks!



- 2 types
  - The goal is to stack a single tower (like the previous type) but the blocks are always required to be in lexicographic order (Ex: A is on top of B, B on top of C, C on top of D etc.)
  - Unstack all blocks from a single stack (random ordering within the stack) and put them on the table.
- 2 examples with COT annotation
- Across 4 to 15 blocks
- Results:
  - **Stacking: 12/12 (100%)**
  - **Unstacking: 59/60 (98.3%)**
    - For 1 instance (10-block) it doesn't execute the pattern completely
- This is actually not surprising as LLMs are great at mimicking patterns!

<u>An example in the prompt (Trivial Stacking)</u>

[STATEMENT]
As initial conditions I have that, Block A is clear, Block B is clear, Block C is clear, Block A is on the table, Block B is on the table, Block C is on the table and the hand is empty.
My goal is to have that Block A is on top of Block B and Block B is on top of Block C.

My plan is as follows:

[THOUGHTS]
1. I look at the goal and stack the tower I need, starting from the bottom.
Block C is on the table and Block C is clear.
Block B is on the table and Block B is clear, so I pick it up.
Then, I stack Block B on top of Block C. The first goal condition is satisfied.
Block A is on the table and Block A is clear, so I pick it up.
Then, I stack Block A on top of Block B. The second goal condition is satisfied.
So all goal conditions are satisfied.

2. Now, I write down my plan in the required format:

[PLAN]
pick up Block B
stack Block B on top of Block C
pick up Block A
stack Block A on top of Block B
[PLAN END]

<u>An example in the prompt (Trivial Unstacking)</u>

[STATEMENT]
As initial conditions I have that, Block B is clear, Block A is on top of Block C, Block B is on top of Block A, the hand is empty and Block C is on the table.
My goal is to have that Block A is on the table, Block B is on the table and Block C is on the table.

My plan is as follows:

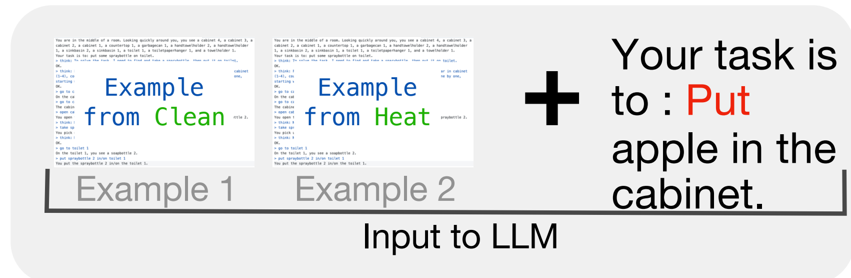
[THOUGHTS]
1. I take apart the tower, and put all the blocks on the table.
Block B is clear.
Block B is on top of Block A, so I unstack it. Then, I put it down on the table.
Now Block A is clear.
Block A is on top of Block C, so I unstack it. Then, I put it down on the table.
Block C is already on the table, so the goal conditions are satisfied.

2. Now, I write down my plan in the required format:

[PLAN]
unstack Block B from on top of Block A
put down Block B
unstack Block A from on top of Block C
put down Block A
[PLAN END]

ReAct Exhibits Similar Inability to generalize

- Requires strong (near syntactic) similarity of example instance to the problem seems to be necessary for ReACT to use the examples!
- Requiring instance-specific examples
- (Our studies also question ReAct's claims about the effectiveness of "Think tag")



ReAct System and Perturbation to Input Examples

Task success rate %, average across 6 tasks : pick, clean, heat, cool, examine, puttwo.
See the gradual drop in performance!

	Base	Replace object names to synonyms	Example Goal location != Query Goal Location	Some examples of different task	All Examples of different task	Examples of each of the tasks	**Unrolling :** Example task is extended in query	**Subtask :** Example task has query as subtask
3.5-turbo	25	1.6	30	12	1.6	14	-	-
3.5-instruct	54	47	42	18	5.2	Context Window Too Short	Drops from 52% to 9%	Drops from 18% to 0%

ReAct Think Tag Claims

1. Think Tag should be interleaved with execution. ✗

Updating Location of think tag to be given once at the beginning (Global). **Performance Improves!**

	Base	Location : Global
3.5-turbo	25	47
3.5-instruct	54	62

2. Think Tag contains Reasoning Trace guidance. ✗

Updating Guidance given as part of think tag. Task success % across 6 classes. **Performance Doesn't Change much!**

	Base	Guidance : Failure	Guidance : Failure based Explanation	Guidance : Magic Incantation	Guidance :Ordering
3.5-turbo	25	47	44	42	28
3.5-instruct	54	62	45	42	43

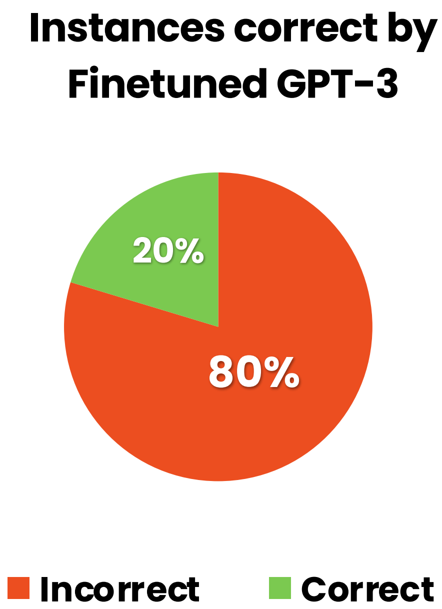
What if we finetuned LLMs with successful plans in the domain?

Finetuning improves the LLM’s generation on a specific distribution.

It doesn’t however guarantee it.

What if we finetuned LLMs with successful plans in the domain?

- What if we further *finetuned* the next word (action) completer with a bunch of correct plans in the domain?
  - This is basically the *supervised finetuning* stage LLMs currently use to make them better at specific domains (e.g. Bloomberg's FinGPT..)
- We prepared a dataset comprising the initial state, goal state, and the respective plan for 1,000 distinct Blocksworld instances.
- By using the default hyperparameters provided by OpenAI and an 80-20 train-validation data split, we carried out the fine-tuning process.
- Finetuned-GPT3 could only solve around **20% (122 out of 600)** of the test set.



Solving Blocksworld: GoFAI vs LLaMAI

GOFAI

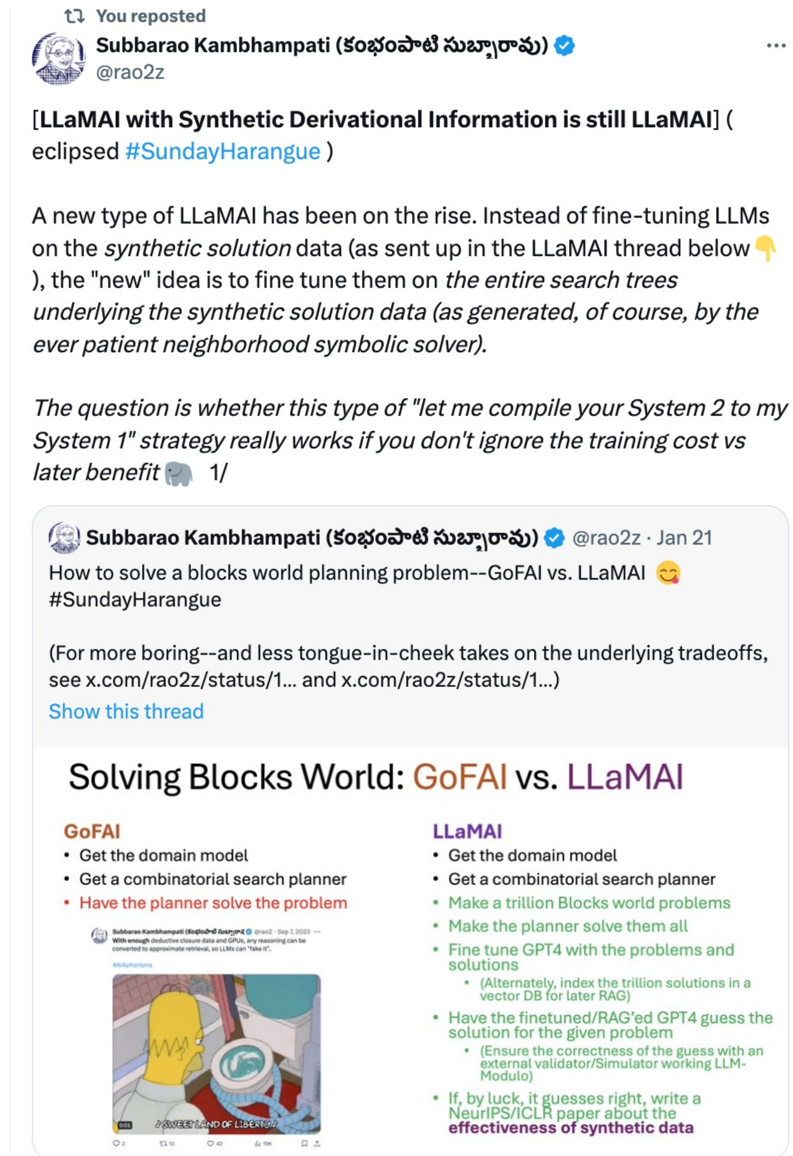
- Get the domain model
- Get a combinatorial search planner
- Have the planner solve the problem

LLaMAI

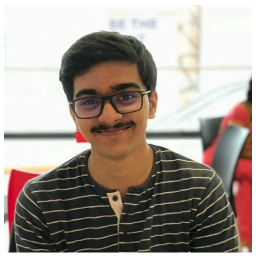
- Get the domain model
- Get a combinatorial search planner
- Make a trillion Blocksworld problems
- Make the planner solve them all
- Finetune GPT4 with the problems and solutions
  - (Alternately, index the trillion solutions in a vector DB for later RAG)
- Have the finetuned/RAG'ed GPT4 guess the solution for the given problem
  - (Ensure the correctness of the guess with an external validator/Simulator working LLM-Modulo)
- If, by luck, it guesses right, write a NeurIPS/ICLR paper about the **effectiveness of synthetic data**

Finetuning with Derivational Traces

- A new twist to fine tuning is to finetune with both solution and the "search/derivational trace" that lead to that solution
  - Supplied of course by the traditional (symbolic) solver
- The question is whether this extended fine tuning generalizes any better of it is still LLaMAI..
  - Little reason to believe it generalizes
  - The evaluation in these papers(*) tends to be quite questionable
    - Claims about "may be optimal"
    - Claims about extending the solving horizon of the base solver



[Lehnert et. al., 2024; Gandhi et. al. 2024]



arXiv > cs > arXiv:2402.08115

Search...

Help | Advanc

Computer Science > Artificial Intelligence

[Submitted on 12 Feb 2024]

On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks

Kaya Stechly, Karthik Valmeekam, Subbarao Kambhampati

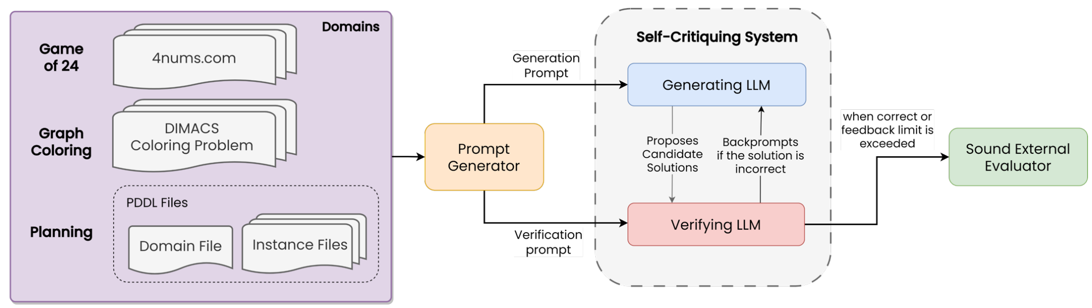
There has been considerable divergence of opinion on the reasoning abilities of Large Language Models (LLMs). While the initial optimism that reasoning might emerge automatically with scale has been tempered thanks to a slew of counterexamples--ranging from multiplication to simple planning--there persists a wide spread belief that LLMs can self-critique and improve their own solutions in an iterative fashion. This belief seemingly rests on the assumption that verification of correctness should be easier than generation--a rather classical argument from computational complexity--which should be irrelevant to LLMs to the extent that what they are doing is approximate retrieval. In this paper, we set out to systematically investigate the effectiveness of iterative prompting in the context of reasoning and planning. We present a principled empirical study of the performance of GPT-4 in three domains: Game of 24, Graph Coloring, and STRIPS planning. We experiment both with the model critiquing its own answers and with an external correct reasoner verifying proposed solutions. In each case, we analyze whether the content of criticisms actually affects bottom line performance, and whether we can ablate elements of the augmented system without losing performance. We observe significant performance collapse with self-critique, significant performance gains with sound external verification, but that the content of critique doesn't matter to the performance of the system. In fact, merely re-prompting with a sound verifier maintains most of the benefits of more involved setups.

Can LLMs self-critique?

The idea that critiquing/verification is easier than generation holds for algorithms that do systematic search.. But not for LLMs that are essentially doing approximate retrieval..

LLMs' self-critiquing abilities

- Three reasoning domains
  - Game of 24, Graph Coloring, Planning
- LLM+LLM System
  - An LLM that generates candidate solutions & an LLM that verifies and critiques it

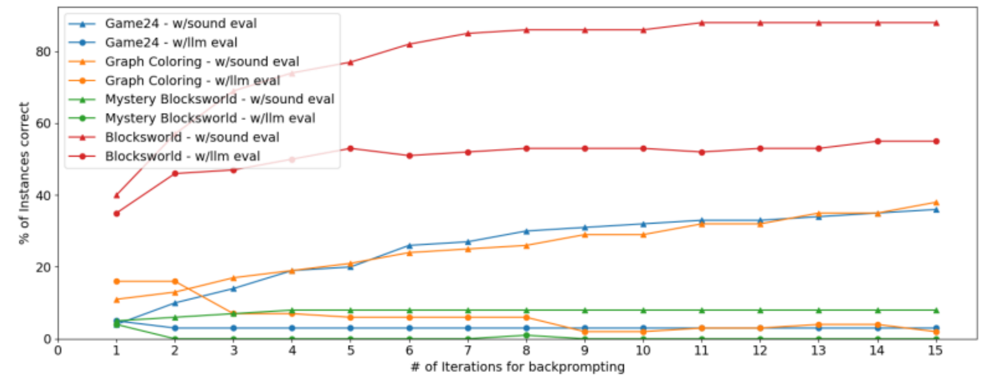


There exist ***formal notions of correctness*** for these domains that allow us to automatically check both the (binary) verification and the critique generated by LLMs.

Such verification is not possible in style-based/qualitative tasks (Eg: writing a good essay, good screenplay etc)

LLMs' self-critiquing abilities

- Standard Prompting
  - A single query is sent to the LLM and whatever it outputs is treated as the final answer
- When this is augmented with the self-critique setup, the performance ***decreases!***
- As the number of back prompts increases, this kind of self-correction consistently degrades output quality.



Domain	S.P.	LLM+LLM	LLM+Sound Critique			Sampling		S.C.
			B.F.	F.E.F	A.E.F	k=15	k=25	k=15
Game of 24	5%	3%	36%	38%	N/A	28%	42%	6%
Graph Coloring	16%	2%	38%	37%	34%	40%	44%	14%
Blocksworld	40%	55%	60%	87%	83%	68%	72%	42%
Mystery Blocksworld	4%	0%	10%	8%	6%	9%	14%	4%

Table 1: **Accuracy across prompting schemes over 100 instances per domain.** S.P.-Standard Prompting. B.F.-Binary Feedback. F.E.F-First Error Feedback, e.g. the first wrong edge, the first mistaken action, or the non-24 evaluation of the proposed expression. A.E.F-All Error Feedback, e.g. every wrong edge, every mistaken action and error. Note that there is no third critique type for Game of 24 due to the simplicity of the domain. We include two examples of sampling, one at 15 samples, the other at 25, to show that completely ablating critique retains the performance increases of critique. We also include S.C.-Self Consistency results, where the most common answer in a pool of 15 is the one that is output by the model, as another comparison point.

LLMs' self-critiquing abilities

- If the LLM were a good verifier, then it would recognize instances which are already right, and thus--at worst--maintain the baseline score.
- The LLM-as-verifier ranges in accuracy depending on the domain, but it maintains **significant false negative rates.**
  - The LLM essentially labels valid solutions to be invalid.
- Also, the solution generator LLM isn't sensitive to varying levels of feedback.
  - In fact, sampling the LLM multiple (k) times for an instance, with a sound verifier in the loop, provides better performance.
    - Connection to Tree of Thoughts..

Domain	**Accuracy**	**F.P.R**	**F.N.R**
Game of 24	87.0% (3567/4100)	10.4% (320/3071)	20.7% (213/1029)
Graph Coloring	72.4% (362/500)	6.5% (25/382)	95.8% (113/118)
Mystery Blocksworld	79.6% (398/500)	0.5% (2/397)	97.09% (100/103)

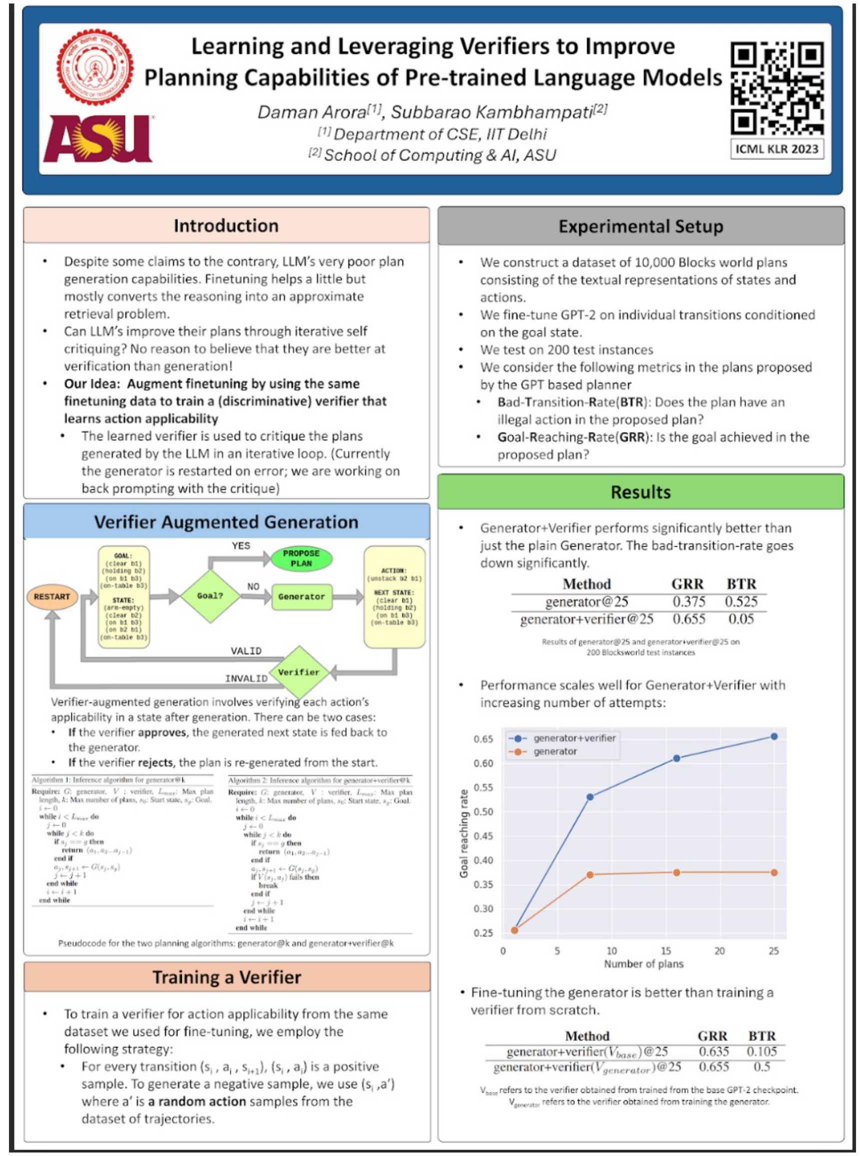
Table 2: **LLM Verification results.** F.P.R. - False Positive Rate, F.N.R - False Negative Rate.

Domain	**Standard Prompting**	**LLM+LLM**	**LLM+Sound Critique**			**Sampling**	
			B.F.	F.E.F	A.E.F	k=15	k=25
Game of 24	5%	3%	36%	38%	N/A	28%	42%
Graph Coloring	16%	2%	38%	37%	34%	40%	44%
Mystery Blocksworld	4%	0%	10%	8%	6%	9%	14%

Table 1: **Accuracy across prompting schemes over 100 instances per domain.** B.F.-Binary Feedback. F.E.F-First Error Feedback, e.g. the first wrong edge, the first mistaken action, or the non-24 evaluation of the proposed expression. A.E.F-All Error Feedback, e.g. every wrong edge, every mistaken action and error. Note that there is no third critique type for Game of 24 due to the simplicity of the domain. We include two examples of sampling, one at 15 samples, the other at 25, to show that completely ablating critique retains the performance increases of critique.

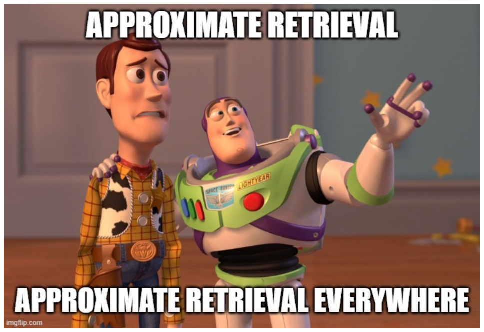
Fine Tuning the Pre-trained model to be both a generator and verifier

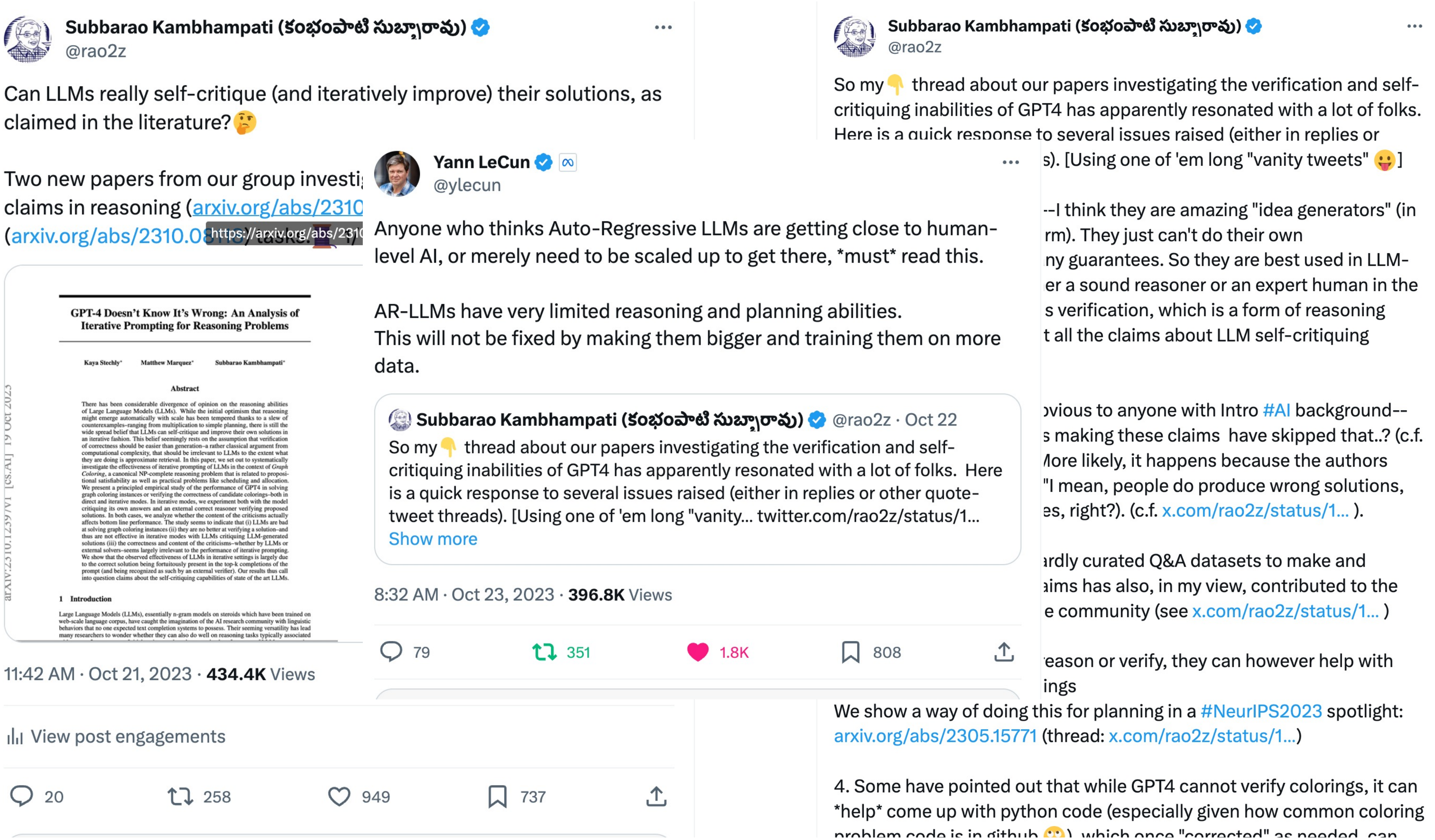
- Start with GPT-2
- [Finetuned generator:] Fine tune GPT-2 as a generator on a corpus of blocks world plans
- [Finetuned Verifier:] Use the same corpus to train a verifier (based off of GPT-2)
- Do Verifier-augmented generation
    - Sort of similar to the back-prompting with VAL (except that the verifier here is also learned from the same corpus)



Why the divide in self-critiquing claims?

- Several other researchers report results that seem to indicate that some form of self-critiquing mode seems to help solving mode. Why?
- **Explicit vs tacit knowledge tasks**
  - It is harder to establish the (poor) quality of LLM critiques in tacit knowledge tasks (like creative writing)
  - In explicit knowledge tasks (like planning, CSP etc) both the verification and critique can be evaluated formally.
- **Approximate retrieval on corrections data informing approximate retrieval on correct data.**
  - For most common use domains (e.g. mine craft, grade school word problems), the training corpora not only contain solution (correct) data, but also corrections data (i.e., the types of normal errors to be found in incorrect solutions).





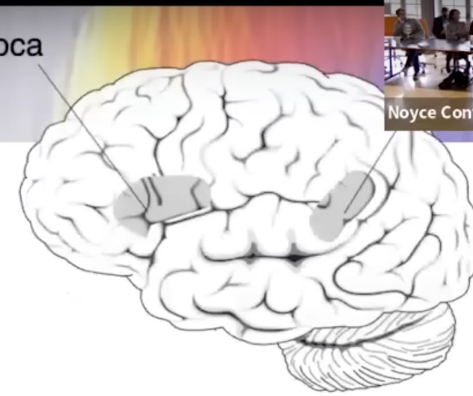
Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z
Can LLMs really self-critique (and iteratively improve) their solutions, as claimed in the literature?🤔
Two new papers from our group investi
claims in reasoning (arxiv.org/abs/2310
(arxiv.org/abs/2310.08
https://arxiv.org/abs/2310
GPT-4 Doesn't Know It's Wrong: An Analysis of Iterative Prompting for Reasoning Problems
Kaya Stechly Matthew Marquez Subbarao Kambhampati
Abstract
1 Introduction
11:42 AM · Oct 21, 2023 · 434.4K Views
View post engagements
20 258 949 737
Yann LeCun
@ylecun
Anyone who thinks Auto-Regressive LLMs are getting close to human-level AI, or merely need to be scaled up to get there, *must* read this.
AR-LLMs have very limited reasoning and planning abilities.
This will not be fixed by making them bigger and training them on more data.
Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Oct 22
So my 👇 thread about our papers investigating the verification and self-critiquing inabilities of GPT4 has apparently resonated with a lot of folks. Here is a quick response to several issues raised (either in replies or other quote-tweet threads). [Using one of 'em long "vanity... twitter.com/rao2z/status/1...
Show more
8:32 AM · Oct 23, 2023 · 396.8K Views
79 351 1.8K 808
Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z
So my 👇 thread about our papers investigating the verification and self-critiquing inabilities of GPT4 has apparently resonated with a lot of folks.
Here is a quick response to several issues raised (either in replies or
s). [Using one of 'em long "vanity tweets" 😛]
--I think they are amazing "idea generators" (in
rm). They just can't do their own
ny guarantees. So they are best used in LLM-
er a sound reasoner or an expert human in the
s verification, which is a form of reasoning
t all the claims about LLM self-critiquing
ovious to anyone with Intro #AI background--
s making these claims have skipped that..? (c.f.
More likely, it happens because the authors
"I mean, people do produce wrong solutions,
es, right?). (c.f. x.com/rao2z/status/1...).
ardly curated Q&A datasets to make and
aims has also, in my view, contributed to the
e community (see x.com/rao2z/status/1...)
eason or verify, they can however help with
ings
We show a way of doing this for planning in a #NeurIPS2023 spotlight: arxiv.org/abs/2305.15771 (thread: x.com/rao2z/status/1...)
4. Some have pointed out that while GPT4 cannot verify colorings, it can *help* come up with python code (especially given how common coloring

Looks Like we showed that LLMs can’t Plan in autonomous modes..?

Yann LeCun: Towards Machines That Can Understand, Reason, & Plan

Limitations of LLMs

- Auto-Regressive LLMs (at best) approximate the functions of the Wernicke and Broca areas in the brain.
- What about the pre-frontal cortex?



ArXiv:2301.06627

DISSOCIATING LANGUAGE AND THOUGHT IN LARGE LANGUAGE MODELS: A COGNITIVE PERSPECTIVE

A PREPRINT

Kyle Mahowald*
The University of Texas at Austin
mahowald@utexas.edu

Anna A. Ivanova*
Massachusetts Institute of Technology
annaiv@mit.edu

Idan A. Blank
University of California Los Angeles
iblank@psych.ucla.edu

Nancy Kanwisher
Massachusetts Institute of Technology
ngk@mit.edu

Joshua B. Tenenbaum
Massachusetts Institute of Technology
jbt@mit.edu

Evelina Fedorenko
Massachusetts Institute of Technology
evelina9@mit.edu

ArXiv:2206.10498

Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change)

Karthik Valmeekam*
School of Computing & AI
Arizona State University, Tempe.
kvalmeek@asu.edu

Alberto Olmo*
School of Computing & AI
Arizona State University, Tempe.
aolmo@asu.edu

Sarath Sreedharan †
Department of Computer Science,
Colorado State University, Fort Collins.
sarath.sreedharan@colostate.edu

Subbarao Kambhampati
School of Computing & AI
Arizona State University, Tempe.
rao@asu.edu

On the other hand, the literature seems rife with claims of LLM planning abilities..

LEAST-TO-MOST PROMPTING ENABLES COMPLEX REASONING IN LARGE LANGUAGE MODELS

Denny Zhou[†*]
Dale Schuurma
[†]Google Researc

Chain-of-Symbol Prompting Elicits Planning in Large Language Models

Hanxu Hu[1*] **Hongyuan Lu**[2*†] **Huajian Zhang**[3] **Wai Lam**[2] **Yue Zhang**[1]
[1]Westlake University [2]The Chinese University of Hong Kong [3]University of Edinburgh
{huhanxu,zhangyue}@westlake.edu.cn
{hylu,wlam}@se.cuhk.edu.hk
v1hzha17@exseed.ed.ac.uk

Abstract

In this paper, we first take the initiative to investigate the performance of LLMs on complex planning tasks that require LLMs to understand a virtual spatial environment simulated via natural language and act correspondingly in text. We propose a benchmark named Natural Language Planning and Action (Natala) composed of a set of novel tasks: Brick World, NLVR-based Manipulations, and Natural Language Navigation. We found that current popular LLMs such as ChatGPT still lack abilities in complex planning. This arises a question – do the LLMs have a good understanding of the environments described in natural language, or maybe other alternatives such as symbolic representations are neater and hence better to be understood by LLMs? To this end, we propose a novel method called CoS (**C**hain-**o**f-**S**ymbol Prompting) that represents the complex environments with condensed symbolic spatial representations during the chained intermediate thinking steps. CoS is easy to use and does not need additional training on LLMs. Extensive experiments indicate that CoS clearly surpasses the performance of the Chain-of-Thought (CoT) Prompting in all three planning tasks with even fewer tokens used in the inputs compared with CoT. The performance gain is strong, by up to 60.8% accuracy (from 31.8% to 92.6%) on Brick World for ChatGPT. CoS also reduces the number of tokens in the prompt obviously, by up to 65.8% of the tokens (from 407 to 139) for the intermediate steps from demonstrations on Brick World. Code and data available at: `https://github.com/hanxuhu/chain-of-symbol-planning`

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

Wenlong Hu
UC Berkel

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima
The University of Tokyo
t.kojima@weblab.t.u-tokyo.ac.jp

Shixiang Shane Gu
Google Research, Brain Team

Machel Reid
Google Research*

Yutaka Matsuo
The University of Tokyo

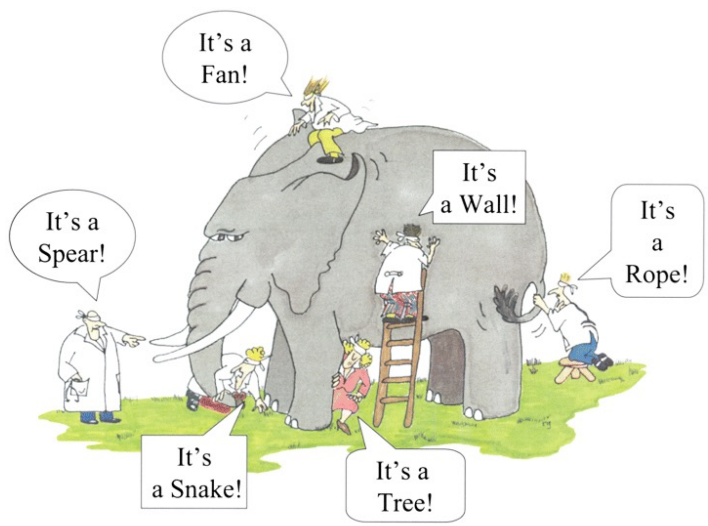
Yusuke Iwasawa
The University of Tokyo

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 often attributed to LLMs' ability for few-shot learning, we show that LLMs are decent *zero-shot* reasoners by simply adding "Let's think step by step" before each answer. Experimental results demonstrate that our Zero-shot-CoT, using the same single prompt template, significantly outperforms zero-shot LLM performances on diverse benchmark reasoning tasks including arithmetics (MultiArith, GSM8K, AQUA-RAT, SVAMP), symbolic reasoning (Last Letter, Coin Flip), and other logical reasoning tasks (Date Understanding, Tracking Shuffled Objects), without any hand-crafted few-shot examples, e.g. increasing the accuracy on MultiArith from 17.7% to 78.7% and GSM8K from 10.4% to 40.7% with large-scale InstructGPT model (text-davinci-002), as well as similar magnitudes of improvements with another off-the-shelf large model, 540B parameter PaLM. The versatility of this single prompt across very diverse reasoning tasks hints at untapped and understudied fundamental *zero-shot* capabilities of LLMs, suggesting high-level, multi-task broad cognitive capabilities may be extracted by simple prompting. We hope our work not only serves as the minimal strongest zero-shot baseline for the challenging reasoning benchmarks, but also highlights the importance of carefully exploring and analyzing the enormous zero-shot knowledge hidden inside LLMs before crafting finetuning datasets or few-shot exemplars.

Why this divide?

Answer: Misunderstandings about what planning involves



Don't attribute to malice that which is adequately explained by ignorance

What Planning is & What LLMs are good at..

Planning (as used in common parlance) involves

- Planning knowledge
  - Actions, preconditions and effects
  - General Recipes: Task reduction schemata (e.g. HTN planning)
  - Old examples: Case libraries
- Plan generation/verification techniques
  - Interaction analysis/resolution
  - Plan merging techniques
  - Plan modification techniques

LLMs accept any planning problem—even if it not expressible in PDDL standard—and they don't give any correctness guarantees.

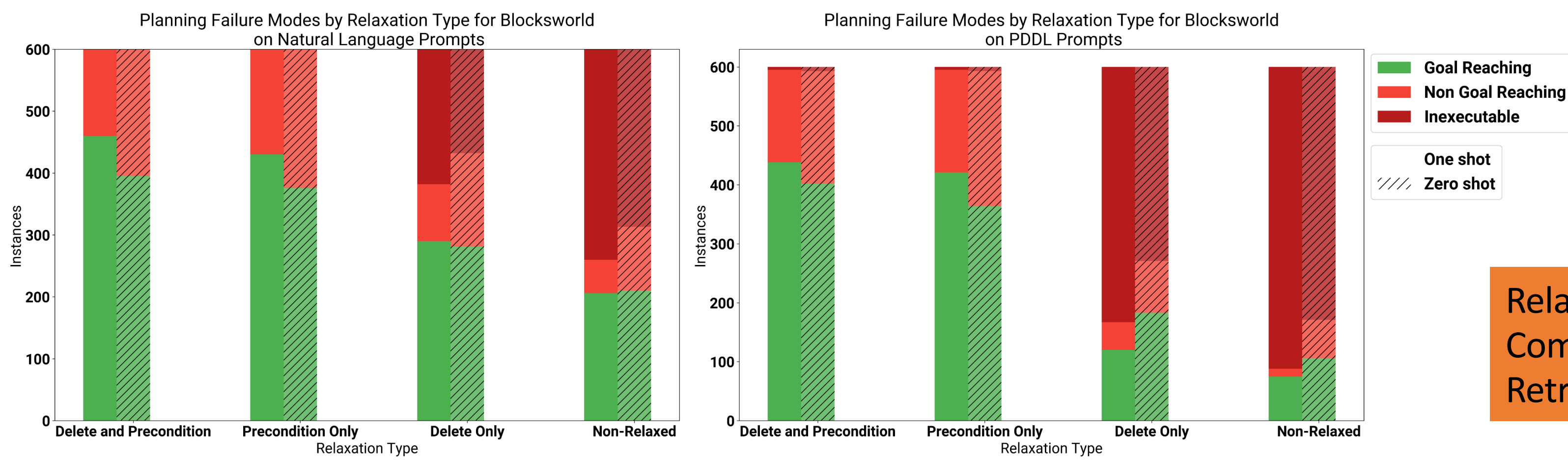
AI Planners will give formal guarantees, but only accept problems expressible in their language.

Contrasting what AI Planning & LLMs bring to the table

- AI Planning (aka ICAPS planning) assumes that the planning knowledge is given up front, and focuses generation and verification techniques
  - Emphasis on guaranteeing completeness/correctness of the plans w.r.t. the model
    - By and large the common paradigm—although there have been occasional mutinies
      - Model-Lite Planning approaches
- LLMs, trained as they are on everything ever put on the web, have a kind of "*approximate omniscience*". This helps them spit out actions, recipes, or cases
  - But they lack the ability to stitch the recipes together to ensure that there is no actually interaction free!

Are LLMs better at planning if there are no subgoal interactions?

- Relaxed assessment of GPT-4 plans
  - Delete relaxation – Ignoring the delete conditions of all actions
  - Precondition relaxation – Ignoring the preconditions of all actions
- Even in the most lenient assessment mode (Delete+Precondition relaxation) there are still plans (~25%) that are not goal reaching.



Relaxations improve Compositionality via Retrieval..

Then how come LLMs are trumpeted as doing planning?

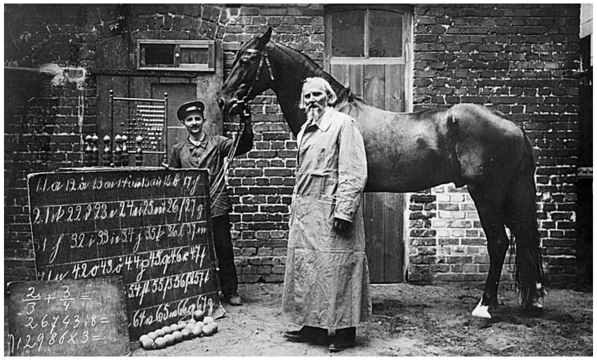
Approximate retrieval of Plans ≠ Planning

- Most cases where LLMs are claimed to generate executable plans, on closer examination, turn out to be cases where LLMs are getting by with the "*generate approximate recipes*" step
  - Generate approximate recipes/cases (for common sense domains)
    - e.g. wedding plans
  - Convert tasks into (approximate) task reduction schemas
    - Perhaps written out as "programs" (e.g. Code as Policies..)
      - (SHOP2 schemas were already pseudo lisp code—if only written by humans)
  - LLM-HTN and LLM-CBR differ from HTN and CBR in that they generate the task-reduction schemas or the cases on demand

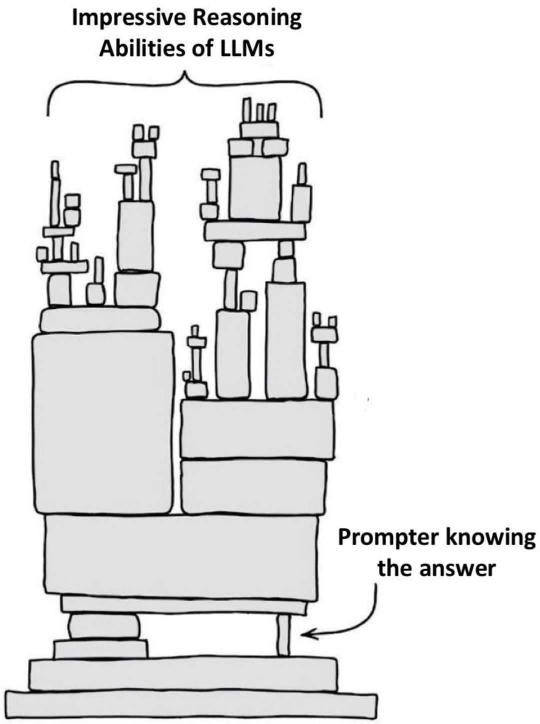
- And the interaction resolution/search part is
  - either *pushed under the rug*
    - Consider "high level" plans like "wedding plans" for which there are enough generic recipes available in the training set, and are described at a sufficiently high level of abstraction, and the execution issues are left to the user's imagination
      - E.g. n-stack blocks world problems with n-1 blocks in the right configuration already!
  - or *has been pawed off to human prompters who are required to give "hints" to the LLM to come up with plan variants that are (more) correct*
    - Note that here the human is essentially playing the role of an external verifier & critic
      - In cases where the humans are end users not well versed with all details of the domain, they can be faulty verifiers

There is also the "Confusing acting with planning" issue

Back-Prompting by Humans (..and the Clever Hans peril..)



- Humans doing the verification & giving helpful prompts to the LLM)
  - Okay when the humans *know* the domain and can correct the plan (with some guarantees)
    - Okay for "this essay looks good enough" kind of critiquing
    - But for planning, with end users not aware of the domain physics, the plans that humans are happy with may still not be actually executable
  - When humans know the correct answer (plan) there is also the very significant possibility of Clever Hans effect
    - Humans unwittingly/unknowingly/non-deliberately giving important hints



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z

Our new paper generalizing the chain, circle and graph of thought prompting strategies--that unleashes the hidden power of LLMs (and graduate students). Hope @_akhaliq picks it up.. 🤞

Forest of Jumbled Thoughts Prompting: An Ultra General Way to use LLMs for Solving Planning, Reasoning, World Peace and Climate Change Tasks

Subbarao Kambhampati
School of Computing & AI
Arizona State University, Tempe.
rao@asu.edu

Abstract

Intrigued by the claims of emergent planning and reasoning capabilities in LLMs, especially in the presence of bright AI graduate students, we have set out to develop the ultimate prompting technique. Our aim is to generalize the chain of thought, circle of thought, tree of thought and graph of thought prompting techniques to a whole another plane. Our "**Forest of Jumbled Thoughts Prompting**" (FJTP) technique is very general, and only requires repeatedly browbeating the LLM to do better by nudging it towards the correct answer. In our experiments on GPT4.5 (that we had got early access to, thanks to our recent investment in OpenAI), we show that our FJTP technique works like a (slow) charm on a variety of planning, reasoning, world peace and climate change tasks. We prove, by reduction to Rube Goldberg Machines, that the FJTP eventually makes LLM "solve" any problem for which the prompting graduate students know the answer. Our proof is general and only assumes an abundant budget for GPT4 API access (or, alternately, co-authors with free access to Palm). The underlying back-to-the-basics "system 2" search that FJTP induces avoids any GOFAI search technology that may need access to things other than LLMs and graduate students. We further show that the solutions that the LLM produces are *exactly the ones the grad students prompt it to produce*–thus ensuring the interpretability and explainability of the solutions generated. We speculate that the awe-inspiring generality of this FJTP prompting technique will eventually make LLMs overcome even their dreaded fear of numbers–and allow them to do arithmetic, thus obviating the need for those *costly* calculators.

5:18 PM · May 19, 2023 · **46.4K** Views

View Tweet analytics

27 Retweets **3** Quotes **161** Likes **89** Bookmarks

Intrigued by the claims of emergent planning and reasoning capabilities in LLMs, especially in the presence of bright AI graduate students, we have set out to develop the ultimate prompting technique. Our aim is to generalize the chain of thought, circle of thought, tree of thought and graph of thought prompting techniques to a whole another plane. Our "**Forest of Jumbled Thoughts Prompting**" (FJTP) technique is very general, and only requires repeatedly browbeating the LLM to do better by nudging it towards the correct answer. In our experiments on GPT4.5 (that we had got early access to, thanks to our recent investment in OpenAI), we show that our FJTP technique works like a (slow) charm on a variety of planning, reasoning, world peace and climate change tasks. We prove, by reduction to Rube Goldberg Machines, that the FJTP eventually makes LLM "solve" any problem for which the prompting graduate students know the answer. Our proof is general and only assumes an abundant budget for GPT4 API access (or, alternately, co-authors with free access to Palm). The underlying back-to-the-basics "system 2" search that FJTP induces avoids any GOFAI search technology that may need access to things other than LLMs and graduate students. We further show that the solutions that the LLM produces are *exactly the ones the grad students prompt it to produce*–thus ensuring the interpretability and explainability of the solutions generated. We speculate that the awe-inspiring generality of this FJTP prompting technique will eventually make LLMs overcome even their dreaded fear of numbers–and allow them to do arithmetic, thus obviating the need for those *costly* calculators.

ReAct, Inner Monologue, ToT Examples

```
(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.
```

```
(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.
```

ReAct

```
===============
Task: Stack all the blocks.

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: False

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: True

Scene: Visible objects are ["brown block", "orange block"]
Scene: Occluded objects are ["purple block"]
Robot action: robot.pick_place("orange block", "brown block")
Successful action: True

Scene: Visible objects are ["orange block"]
Scene: Occluded objects are ["purple block", "brown block"]
Robot action: robot.stop()
STOP

===============
Task: Stack all the blocks.

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: True

Scene: Visible objects are ["brown block"]
Scene: Occluded objects are ["purple block"]
Robot action: robot.stop()
STOP

===============
```

Inner Monologue

	Game of 24	Creative Writing	5x5 Crosswords
Input	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1. presented;..)
Output	An equation to reach 24 (13-9)*(10-4)=24	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: SHOWN; WIRRA; AVAIL; ...
Thoughts	3 intermediate equations (13-9=4 (left 4,4,10); 10-4=6 (left 4,6); 4*6=24)	A short writing plan (1. Introduce a book that connects...)	Words to fill in for clues: (h1. shown; v5. naled; ...)
#ToT steps	3	1	5-10 (variable)

Table 1: Task overview. Input, output, thought examples are in blue.

Tree of Thoughts

Most of the 'planning' problems that these works look at don't require interaction resolution, or they depend on explicit external help/programming to handle the interactions.

Shoddy guidebooks, promoted with deceptive reviews, have flooded Amazon in recent months.

Travel Planning Benchmark

- New benchmark for travel planning proposed in Feb 2024
- Three different types of constraints
  - Environment constraints
  - Common-sense constraints
  - Hard constraints
- **GPT-4-Turbo could solve only 0.6% (out of 1000 queries)**
- Not surprising! We show that LLM's can't even stack blocks correctly, there's surely no hope for travel planning that has lots of constraints!!

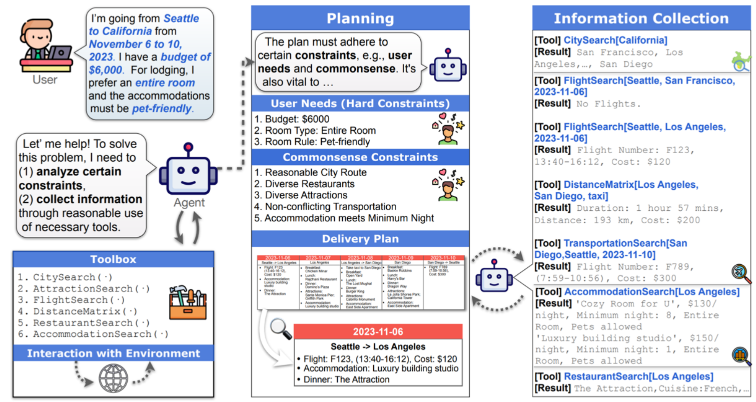


Figure 1. Overview of TravelPlanner. Given a query, language agents are tasked with employing various search tools to gather information. Based on the collected information, language agents are expected to deliver a plan that not only satisfies the user's needs specified in the query but also adheres to commonsense constraints.

Table 1. Constraint description. The environment constraints are manifested through the feedback received from the environment, assessing whether the language agent can adjust its plan appropriately. The commonsense constraints and hard constraints are evaluated based on how well the language agent's plan aligns with these specific criteria.

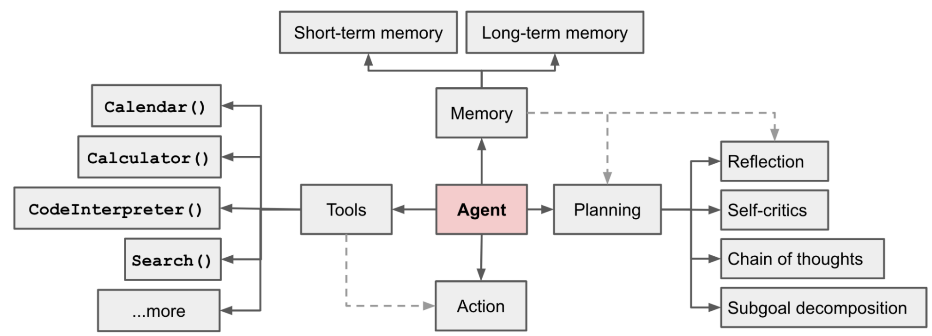
Constraint	Description
	Environment Constraint
Unavailable Transportation	There is no available flight or driving information between the two cities.
Unavailable Attractions	There is no available attraction information in the queried city.
	Commonsense Constraint
Within Sandbox	All information in the plan must be within the closed sandbox; otherwise, it will be considered a hallucination.
Complete Information	No key information should be left out of the plan, such as the lack of accommodation during travel.
Within Current City	All scheduled activities for the day must be located within that day's city(s).
Reasonable City Route	Changes in cities during the trip must be reasonable.
Diverse Restaurants	Restaurant choices should not be repeated throughout the trip.
Diverse Attractions	Attraction choices should not be repeated throughout the trip.
Non-conf. Transportation	Transportation choices within the trip must be reasonable. For example, having both "self-driving" and "flight" would be considered a conflict.
Minimum Nights Stay	The number of consecutive days spent in a specific accommodation during the trip must meet the corresponding required minimum number of nights' stay.
	Hard Constraint
Budget	The total budget of the trip.
Room Rule	Room rules include "No parties", "No smoking", "No children under 10", "No pets", and "No visitors".
Room Type	Room types include "Entire Room", "Private Room", "Shared Room", and "No Shared Room".
Cuisine	Cuisines include "Chinese", "American", "Italian", "Mexican", "Indian", "Mediterranean", and "French".
Transportation	Transportation options include "No flight" and "No self-driving".

Xie, J., Zhang, K., Chen, J., Zhu, T., Lou, R., Tian, Y., ... & Su, Y. (2024). TravelPlanner: A Benchmark for Real-World Planning with Language Agents. *arXiv preprint arXiv:2402.01622*.

Acting vs. Planning: The Agentic LLM Goldrush

- LLMs can obviously be used to invoke external actions
- Think "Webservice Orchestration Frameworks" which allow you to write your own "agents"
  - LLM as the core controller of external components
    - Which in turn is controlled by human prompting
    - Safety issues include both safety of the outside components and safety of the prompt-based control of LLMs
- LLMs can't themselves be expected to "plan" this orchestration!
  - The actual orchestration is done with human help ("language" programming)
    - The "planning" part is basically pipelining the right external services – and is done **with human help**
  - One core external service they all use is "external memory" to write into and retrieve
    - Because LLMs themselves have no memory beyond their context window.
  - Think L2/L3 rather than L5 automation..

The Agentification



Allowing LLMs to make their own "plans" to invoke external services would be rife with safety concerns!

(Think having a gun lying around in a home with a toddler..)

Weng, Lilian. (Jun 2023). LLM-powered Autonomous Agents". Lil'Log. https://lilianweng.github.io/posts/2023-06-23-agent/.

Doesn't Co-Pilot for Code show that LLMs can Plan?

- Co-Pilot has humans in the loop
  - The incremental interpreters can direct people's attention to syntax errors
- Github and General Web are quite different as training corpora
  - People don't put their non-working code on github; general web has 4Chan!

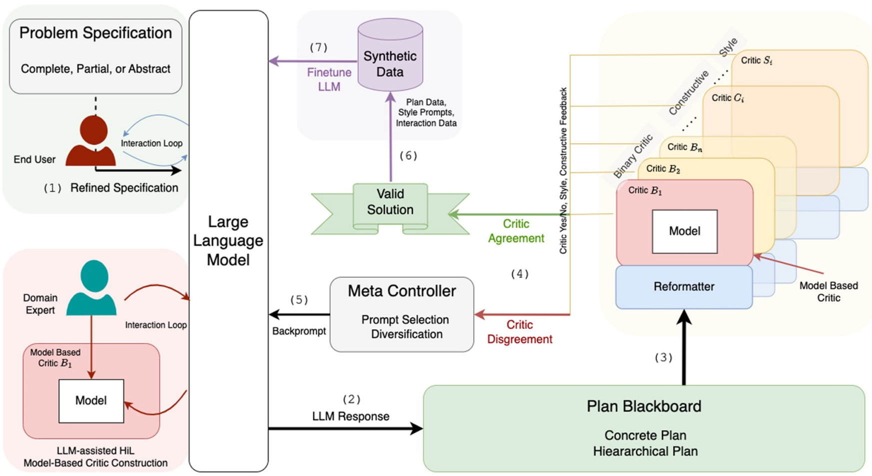
Ability to approximately retrieve code segments

$\neq$

Ability to reason and plan

Talk Overview

- Part 1
  - Evaluating LLM Planning capabilities in Autonomous mode, including effect of
    - Prompting strategies (including Chain-of-thought)
    - Fine Tuning
    - Self Verification
  - Understanding the contradictory claims in the literature
- **Part 2**
  - Sane roles of LLMs in Planning (with LLM-Modulo frameworks)
    - LLMs as heuristics, LLMs as candidate generators
    - Back prompting from external verifiers
    - LLMs as sources of domain models (with humans in the loop)
    - LLMs as format changers/specification elaborators1
- **Part 3**
  - Some thoughts on why people tend to ascribe planning/reasoning abilities to LLMs

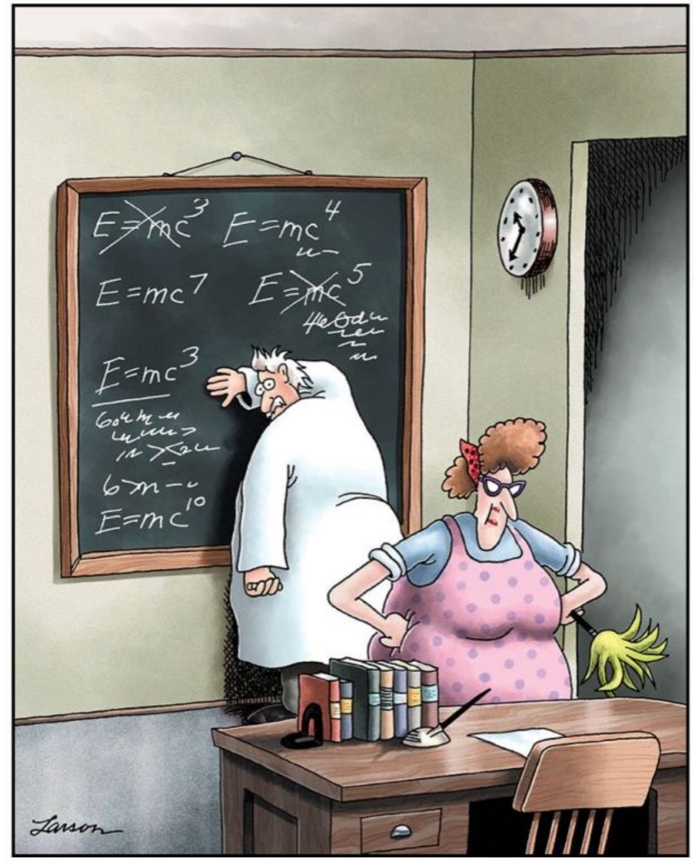


Can LLMs help in Planning at all?

LLMs as Idea Generators ("Muses")

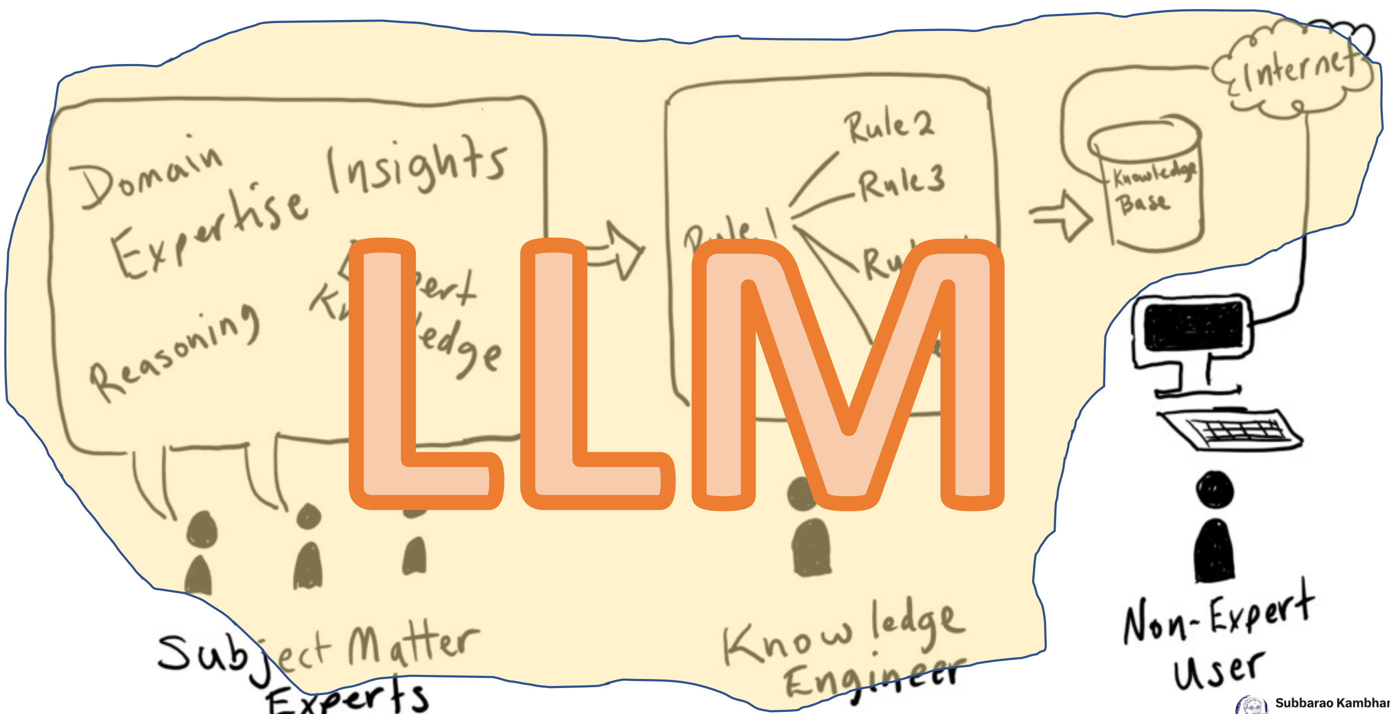
- "I get many ideas, and I throw away the bad ones"
  - Linus Pauling on how he managed to get TWO Nobels

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Sep 3 ···
Replying to @rao2z @liron and @DynamicWebPaige
IMHO, LLM's are impressive *idea generators* for anything--including "reasoning" tasks. But an idea generator is not the same as a sound reasoner. **Fermat** had an idea/conjecture; Wiles spent 20 years and proved it. Neither was subsumed by the other! 3/

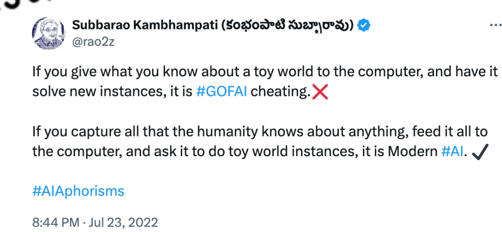


"*Now* that desk looks better. Everything's squared away, yessir, squaaaaaared away."

LLMs as Approximate Knowledge Sources



Avenging Polanyi's Revenge
Everybody was all against knowledge-based systems
But now everyone is effectively doing knowledge-based systems!


Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z

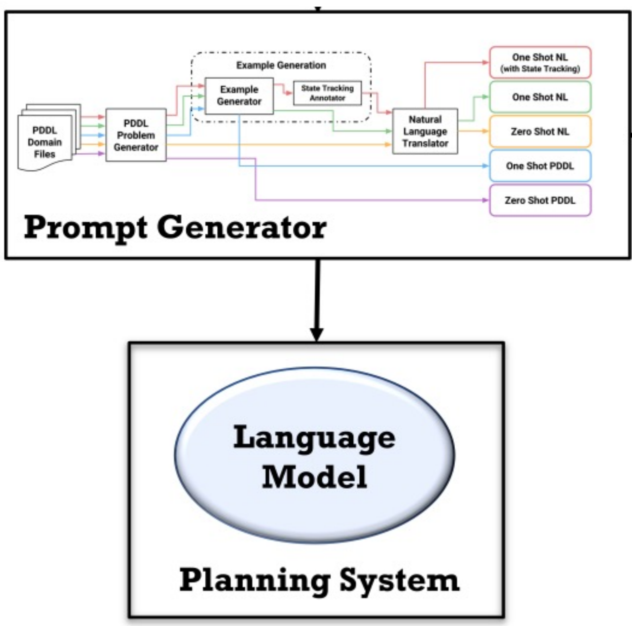
If you give what you know about a toy world to the computer, and have it solve new instances, it is #GOFAI cheating.❌

If you capture all that the humanity knows about anything, feed it all to the computer, and ask it to do toy world instances, it is Modern #AI. ✓

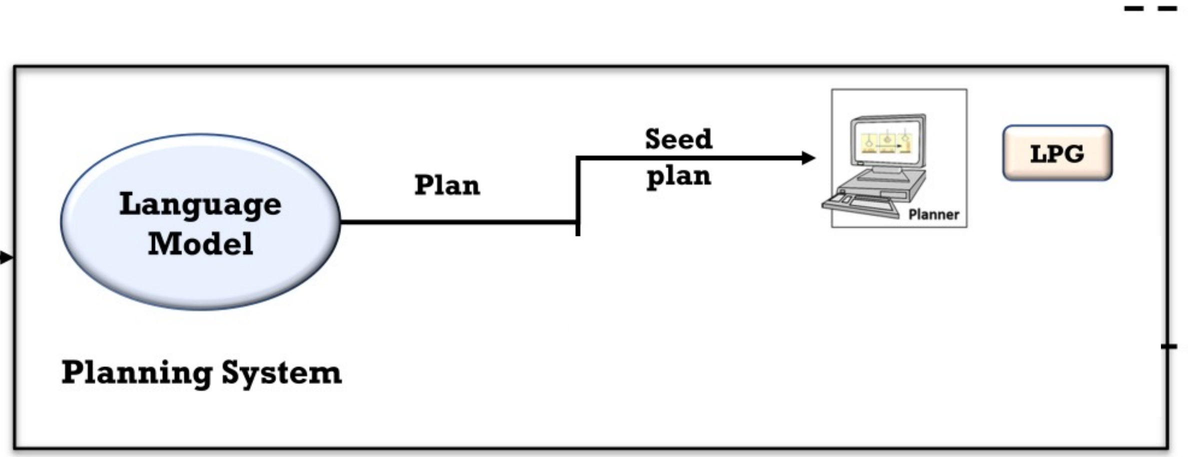
#AIAphorisms

8:44 PM · Jul 23, 2022

Workflow for using LLM’s as Idea Generators (for External Sound Planners)



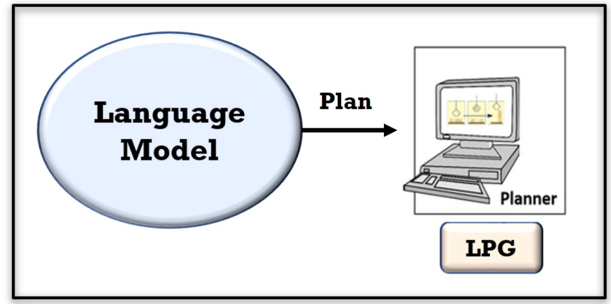
LLM as an autonomous planner



LLM as heuristics to sound planners like LPG

LLM’s have universal high-recall (they will never shut up!), but questionable precision
Automated Planners are guaranteed correct but for planning problems that they can handle

LLMs as heuristics to sound planners



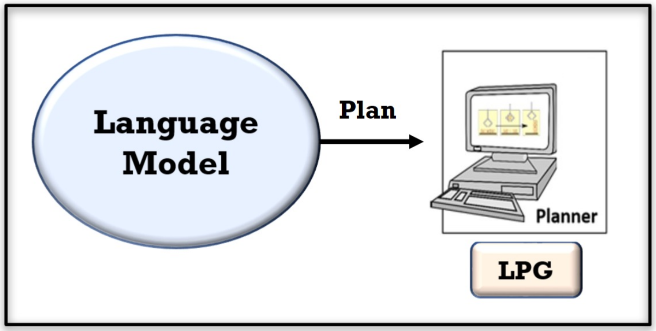
LLM generated plan as a heuristic to a sound planner like LPG

Table 3: Evaluation of GPT-4 and Instruct-GPT3 (I-GPT-3) plans as heuristics for a local search planner LPG, on blocksworld (BW), logistics and mystery blocksworld domains.

Domain	LLM	Avg. Search Steps			Avg. Plan Length			Avg. Lev. Distance
		Empty Seed Plan	Random Seed Plan	LLM Seed Plan	Empty Seed Plan	Random Seed Plan	LLM Seed Plan	
BW	I-GPT-3	15.8	20.07	14.5	8.45	9.62	11.7	7.22
	GPT-4	15.8	20.07	8.9	8.45	9.62	10.76	4.15
Logistics	GPT-4	77.5	144.39	51.3	23.7	32.72	32.24	15.04
Mystery BW	GPT-4	15.8	20.45	16.09	8.45	9.78	11.53	7.77

Connection to Case based Planning

- Note that there is an interesting parallel between this and case based planning systems—which retrieve an old plan most relevant to the current problem and try to modify the plan
  - Modification by domain-specific rules [e.g. CHEF]
  - Modification by domain-independent planners [e.g. PRIAR]
- LLM-CBR is different in that the case is generated ("stitched") on demand
  - ..and LPG is in charge of correcting it



LLM generated plan as a heuristic to a sound planner like LPG

[Submitted on 2 Feb 2024 (v1), last revised 6 Feb 2024 (this version, v2)]

LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, Anil Murthy

There is considerable confusion about the role of Large Language Models (LLMs) in planning and reasoning tasks. On one side are over-optimistic claims that LLMs can indeed do these tasks with just the right prompting or self-verification strategies. On the other side are perhaps over-pessimistic claims that all that LLMs are good for in planning/reasoning tasks are as mere translators of the problem specification from one syntactic format to another, and ship the problem off to external symbolic solvers. In this position paper, we take the view that both these extremes are misguided. We argue that auto-regressive LLMs cannot, by themselves, do planning or self-verification (which is after all a form of reasoning), and shed some light on the reasons for misunderstandings in the literature. We will also argue that LLMs should be viewed as universal approximate knowledge sources that have much more meaningful roles to play in planning/reasoning tasks beyond simple front-end/back-end format translators. We present a vision of {\bf LLM-Modulo Frameworks} that combine the strengths of LLMs with external model-based verifiers in a tighter bi-directional interaction regime. We will show how the models driving the external verifiers themselves can be acquired with the help of LLMs. We will also argue that rather than simply pipelining LLMs and symbolic components, this LLM-Modulo Framework provides a better neuro-symbolic approach that offers tighter integration between LLMs and symbolic components, and allows extending the scope of model-based planning/reasoning regimes towards more flexible knowledge, problem and preference specifications.

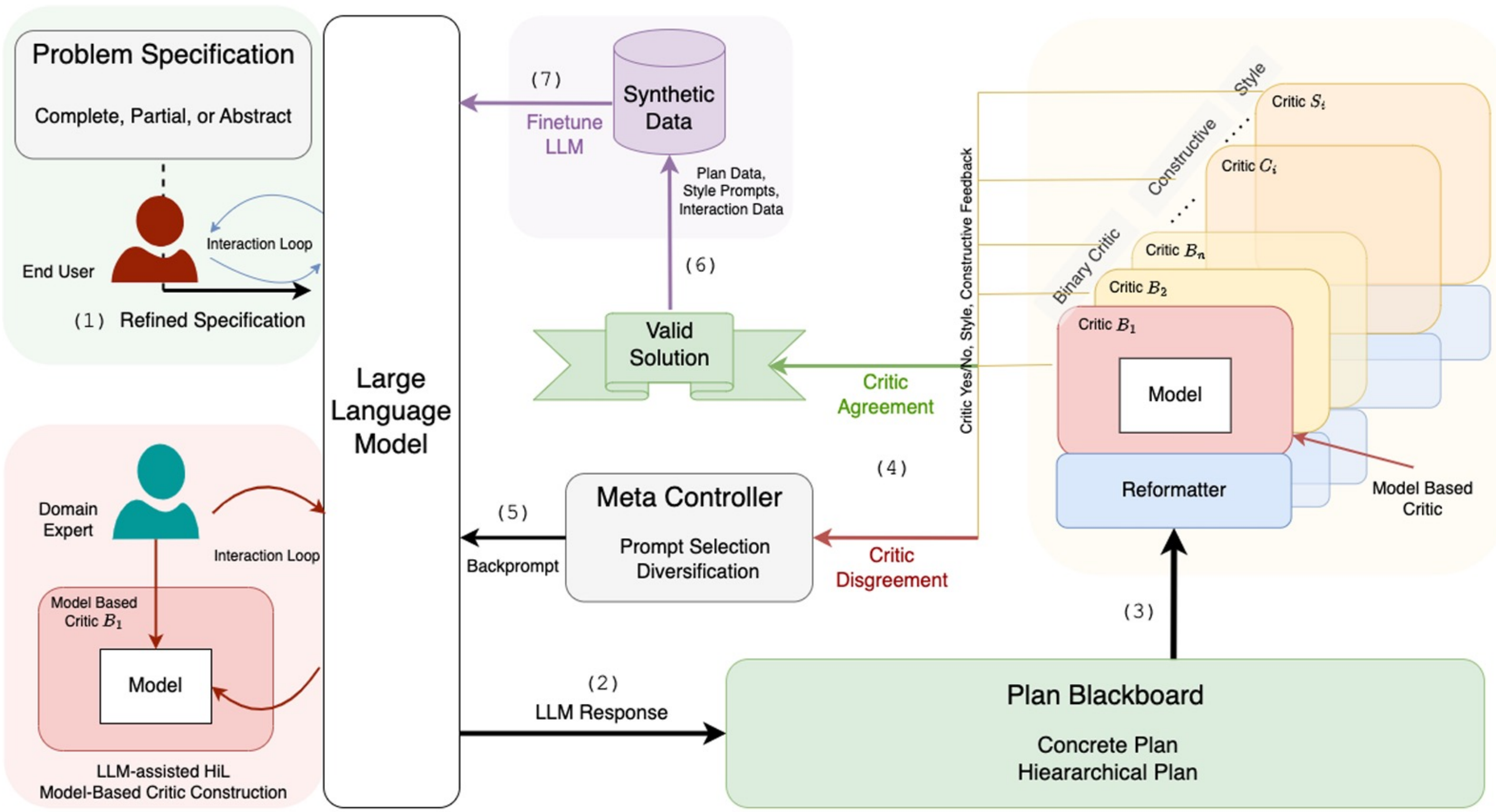
Subjects: **Artificial Intelligence (cs.AI)**; Machine Learning (cs.LG)

Cite as: arXiv:2402.01817 **[cs.AI]**

(or arXiv:2402.01817v2 **[cs.AI]** for this version)

https://doi.org/10.48550/arXiv.2402.01817

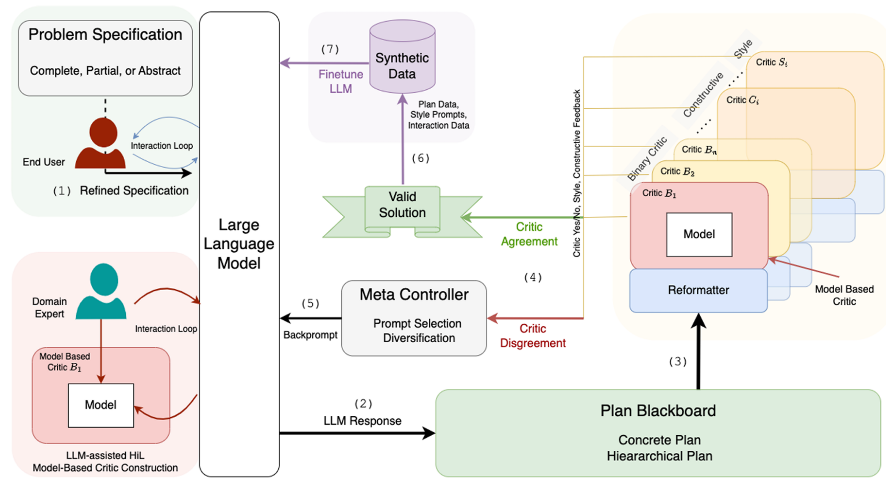
LLM-Modulo: a principled framework for Planning wherein LLMs can play multiple constructive roles



Prefer Verifiers to Solvers!
[Solver ≈ Verifier + Search]

- Although we just saw a way of combining LLMs with external solvers, our *recommendation* is that you interface LLMs with Verifiers/Critics
  - This is why the LLM-Modulo architecture talks about a bank of critics
  - You can have constructive critics and style critics
- With solvers, you are stuck with their expressiveness issues
  - Verifiers, on the other hand, can allow composability, and validating the plan to the extent possible
  - Similar to the "Human Blackboard" architecture used in NASA mission planning..

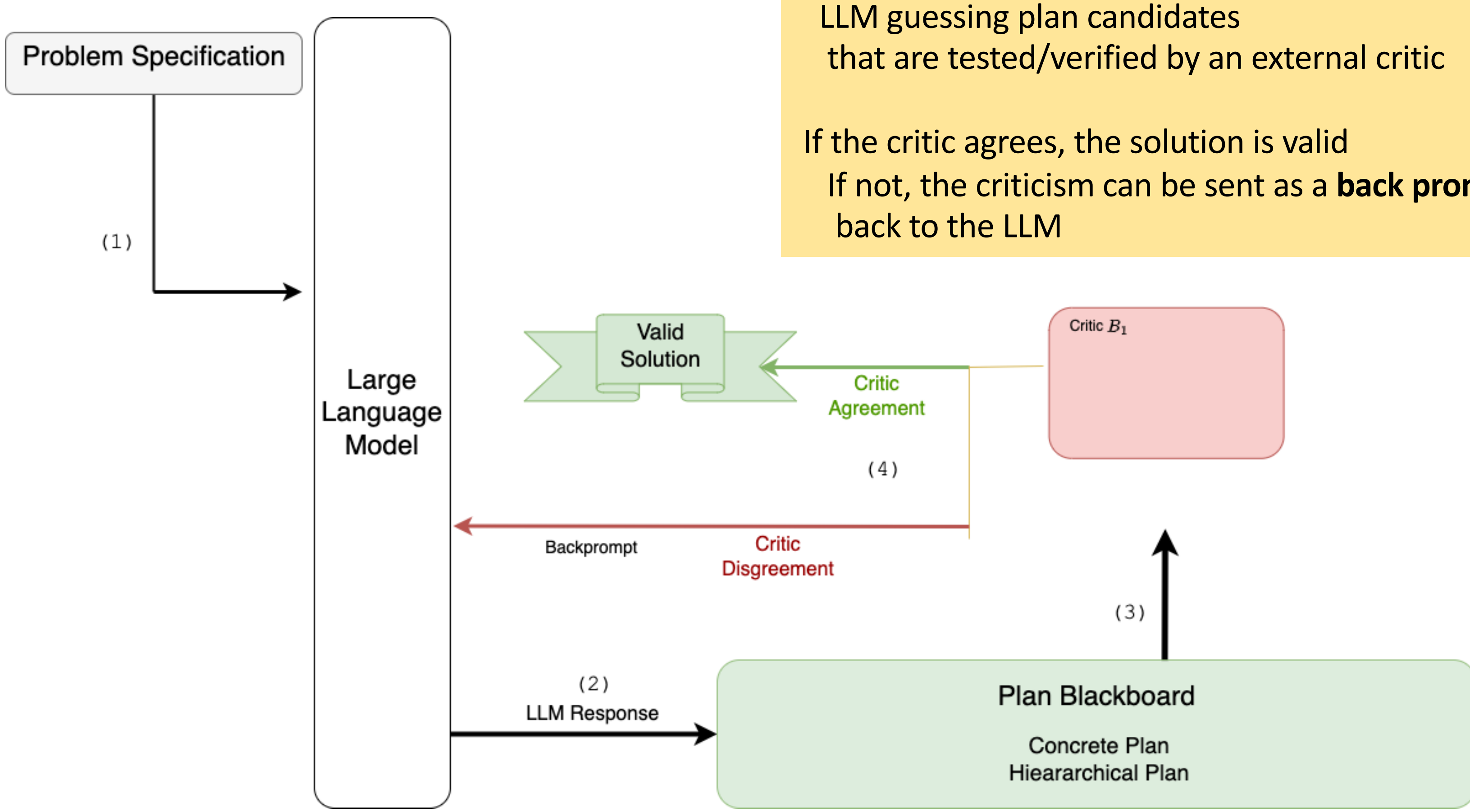
LLM-Modulo: a principled framework for Planning wherein LLMs can play multiple constructive roles



Bare Bones Generate-Test LLM-Modulo with External Verifier

At it's simplest, LLM-Modulo is a loop with
LLM guessing plan candidates
that are tested/verified by an external critic

If the critic agrees, the solution is valid
If not, the criticism can be sent as a **back prompt** back to the LLM



Automated Back-Prompting with External Verifiers

- Preliminary experiments show that back-prompting does improve LLM's ability to produce plans in the Blocks World and Logistics
  - On the average over ~4 feedback rounds
- The performance in the Mystery BW still doesn't improve-– showing that the connection to commonsense domains/terms is critical for LLMs to *fake planning*

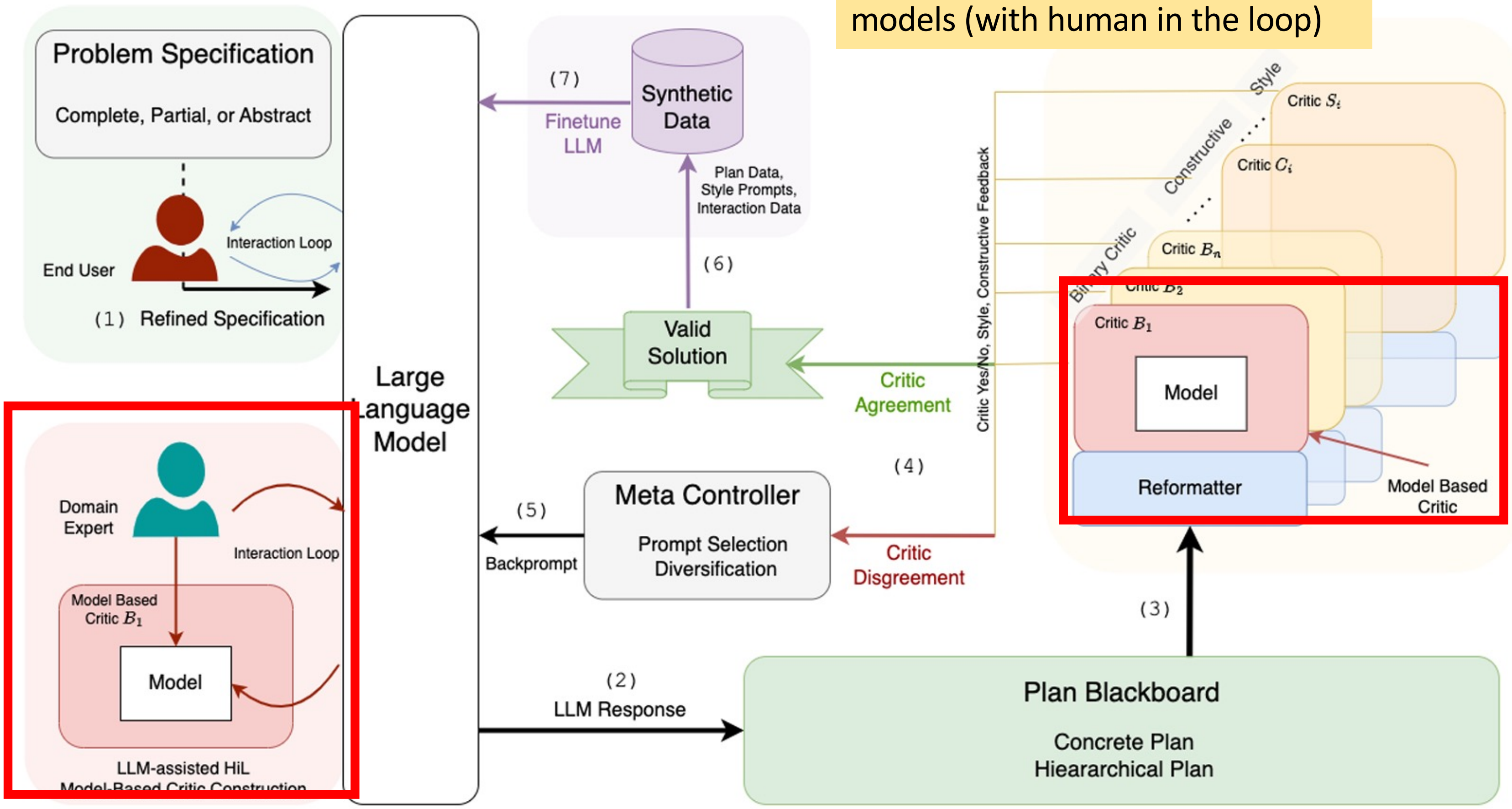
Table 4: GPT4 Performance with Backprompting by VAL [9]. Mystery BW had deceptive disguising. I.C - Instances correct (within 15 feedbacks); A.F.R - Avg. feedback rounds for correct instances.

Domain	I.C	A.F.R
	GPT-4	GPT-4
Blocksworld (BW)	41/50 (82%)	3.68
Logistics	35/50 (70%)	3.31
Mystery BW	5/50 (10%)	7.0

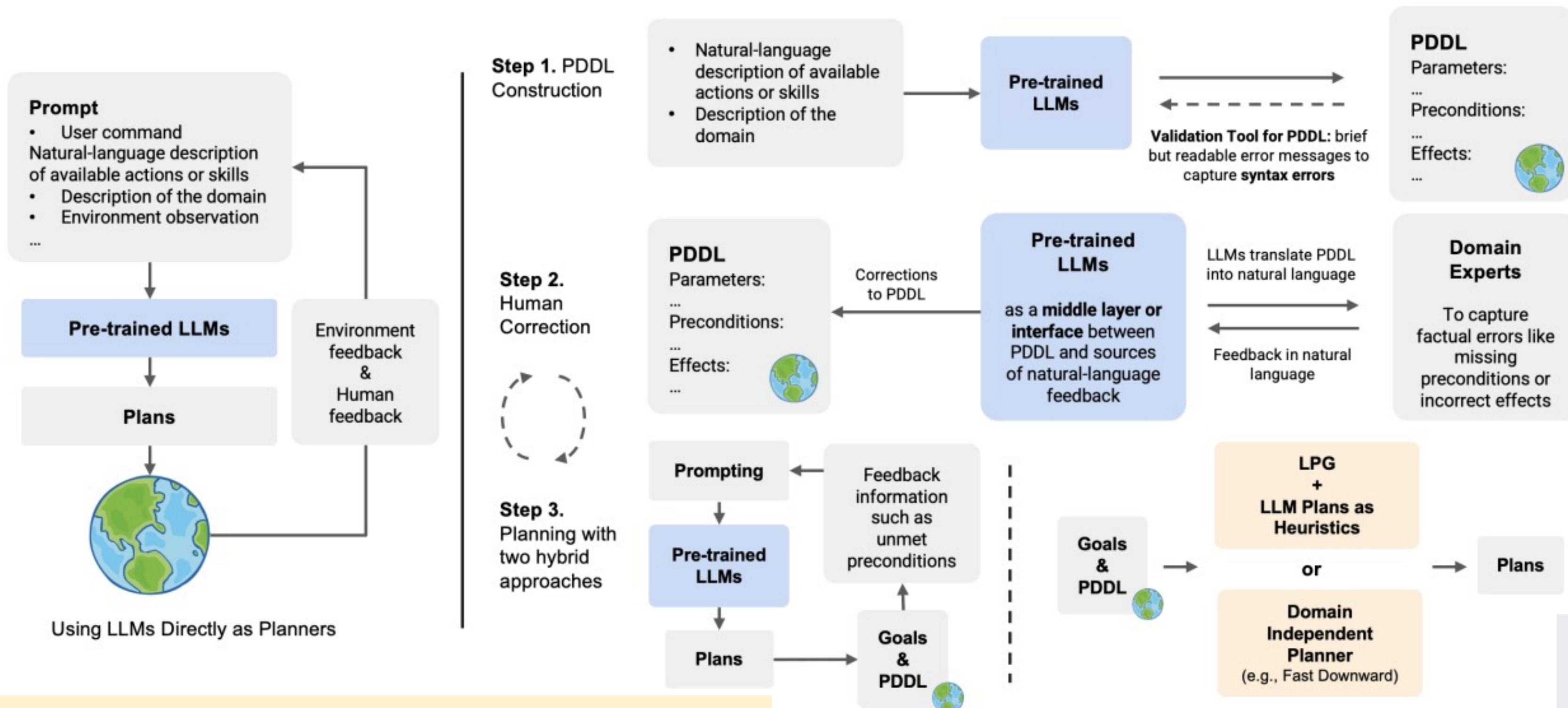
The fact that Mystery BW doesn't improve with Backprompting is *further evidence* that LLMs are *Approximate Retrievers…*

LLMs for Extracting Planning Knowledge

Since LLMs are approximate knowledge sources, they can also be helpful in acquiring domain models (with human in the loop)



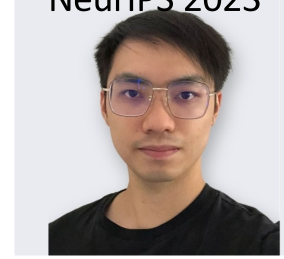
LLMs for constructing domain/world models (Model Co-Pilot)



LLM's have universal high-recall (they will never shut up!), but questionable precision

Automated Planners are guaranteed correct but for planning problems that they can handle

NeurIPS 2023



LLMs for constructing world models

- We utilize LLMs to extract a symbolic representation of the actions in the form of PDDL action models
- This intermediate output can be used with an external domain-independent planner to reliably search for feasible plans, or it can be used to validate and correct "heuristic" plans generated by an LLM planner.
- We also show that LLMs can also serve as an interface between PDDL and any feedback sources that can provide corrective feedback in natural language, such as humans and the PDDL validator in VAL

```
Instructions for the PDDL generation task
You are defining the preconditions and effects (represented in PDDL format) of an AI agent's
↪  actions. Information about the AI agent will be provided in the domain description ...
One or two examples from other domains for illustrating the input and output formats
Here are two examples from the classical BlocksWorld domain for demonstrating the output format.
...
Here is the task.
A natural language description of the domain
Domain information: The AI agent here is a household robot that can navigate to various large and
↪  normally immovable furniture pieces or appliances in the house to carry out household tasks
↪  ...
A natural language description of the action
Action: This action enables the robot to toggle small appliances (like humidifiers and light
↪  bulbs) which are toggleable to switch them on ...
The dynamically updated list of predicates
You can create and define new predicates, but you may also reuse the following predicates:
1. (robot-at ?r - robot ?f - furnitureAppliance): true if the robot ?r is at the furniture or
↪  appliance ?f
2. (object-in-on ?o - householdObject ?f - furnitureAppliance): true if the object ?o is in or on
↪  the furniture or appliance ?f
...
Parameters:
---------------------------------------
The LLM:
...
2. ?o - householdObject: the small appliance to be toggled on
...

Preconditions:
(and
    ...
    (not (appliance-on ?o))
)

Effects:
(and
    (appliance-on ?o)
)

New Predicates:
1. (appliance-on ?o - householdObject): true if the small appliance ?o is switched on
```

Step 1. PDDL Construction

Incrementally construct the domain model: our algorithm generates PDDL models for each action separately, one at a time, by iterating over the set of actions.

Newly defined predicates are appended to the list of extracted predicates

```
Instructions for the PDDL generation task
You are defining the preconditions and effects (represented in PDDL format) of an AI agent's
↪ actions. Information about the AI agent will be provided in the domain description ...
One or two examples from other domains for illustrating the input and output formats
Here are two examples from the classical BlocksWorld domain for demonstrating the output format.
...
Here is the task.
A natural language description of the domain
Domain information: The AI agent here is a household robot that can navigate to various large and
↪ normally immovable furniture pieces or appliances in the house to carry out household tasks
↪ ...
A natural language description of the action
Action: This action enables the robot to toggle small appliances (like humidifiers and light
↪ bulbs) which are toggleable to switch them on ...
The dynamically updated list of predicates
You can create and define new predicates, but you may also reuse the following predicates:
1. (robot-at ?r - robot ?f - furnitureAppliance): true if the robot ?r is at the furniture or
↪ appliance ?f
2. (object-in-on ?o - householdObject ?f - furnitureAppliance): true if the object ?o is in or on
↪ the furniture or appliance ?f
...
Parameters:
---------------------------------------
The LLM output
1. ?r - robot: the robot performing the action
2. ?o - householdObject: the small appliance to be toggled on
3. ?f - furnitureAppliance: the furniture or appliance where the small appliance is located

Preconditions:
(and
    (robot-at ?r ?f)
    (object-in-on ?o ?f)
    (robot-gripper-empty ?r)
    (not (appliance-on ?o))
)

Effects:
(and
    (appliance-on ?o)
)

New Predicates:
1. (appliance-on ?o - householdObject): true if the small appliance ?o is switched on
```

Every time a new predicate is defined, the LLM is required to give the natural language description of it.

Step 2. Correcting PDDL

Sources of corrective feedback

- Syntax error: PDDL validations like the one in VAL
- Factual error: human domain experts

Corrective feedback is integrated by replaying and continuing the PDDL-construction dialogue.

```
The LLM:
The generated PDDL model for "heating food with a pan":
Parameters:
1. ?r - robot: the robot that will heat the food
...
Incorrect object type: stove
4. ?s - stove: the stove burner where the pan is placed

Preconditions:
(and
    ...
    Unnecessary precondition
    (object-pickupable ?f)
)
...
```

Corrective feedback in natural language:

There are some errors in the PDDL model:
1. There is an invalid object type `stove` for the parameter ?s
2. There is an unnecessary precondition "the food to heat is pickup-able"

Please revise the PDDL model (and the list of predicates if needed) to fix the above errors (and other potentially similar errors).

LLMs for constructing world models

- We tested on three domains
  1. Household domain
  2. Logistics
  3. Tyreworld

Domain	# of actions	# of params and literals	# of GPT-4 errors	# of GPT-3.5-Turbo errors
Household	22	271	53	218+
Logistics	6	54	2	38
Tyreworld	13	108	4	94+

Table 1: The number of errors in the domain models produced by the LLMs for each of the domains. A "+" mark indicates that the generated model is excessively noisy, making it challenging to determine an exact number of errors.

Action description

```
This action enables the robot to toggle a small appliances (like humidifiers and light bulbs) to
↪ switch them off. For example, the robot toggles humidifier_2 off, or the robot toggle
↪ light_bulb_1 off.
```

GPT4: Toggle a small appliance off

```
Parameters:
1. ?r - robot: the robot performing the action
2. ?o - householdObject: the small appliance to be toggled off
3. ?f - furnitureAppliance: the furniture or appliance where the small appliance is located

Preconditions:
(and
    (robot-at ?r ?f)
    (object-in-on-furniture ?o ?f)
    (robot-gripper-empty ?r)
    (appliance-on ?o)
)

Effects:
(and
    (not (appliance-on ?o))
)
```

Example from Household domain

Action description

```
This action enables the agent to load a package into a truck. For example, load a package_1 into a
↪ truck_1.
```

GPT-4: Load a package into a truck

```
Parameters:
1. ?p - package: the package to load
2. ?t - truck: the truck to load the package into
3. ?l - location: the location where the package and truck are

Preconditions:
(and
    (package-at ?p ?l)
    (truck-at ?t ?l)
)

Effects:
(and
    (not (package-at ?p ?l))
    (package-in-truck ?p ?t)
)
```

Example from Logistics domain

LLMs for constructing world models

- We tested on three domains
  1. Household domain
  2. Logistics
  3. Tyreworld

Domain	# of actions	# of params and literals	# of GPT-4 errors	# of GPT-3.5-Turbo errors
Household	22	271	53	218+
Logistics	6	54	2	38
			4	94+

; produced by the LLMs for each of the domains.
ssively noisy, making it challenging to determine

- Preliminary results show that LLMs actually do help in acquiring domain models in a semi-automated way
- The acquired domain models can be used either by external AI planners, or in a back-prompting strategy with a verifier operating on the acquired model

Action description

```
This action enables the robot to toggle a sma
↪ switch them off. For example, the robot t
↪ light_bulb_1 off.
```

GPT4: Toggle a small appliance off

```
Parameters:
1. ?r - robot: the robot performing the actic
2. ?o - householdObject: the small appliance
3. ?f - furnitureAppliance: the furniture or

Preconditions:
(and
    (robot-at ?r ?f)
    (object-in-on-furniture ?o ?f)
    (robot-gripper-empty ?r)
    (appliance-on ?o)
)

Effects:
(and
    (not (appliance-on ?o))
)
```

Example from Household domain

```
e into a truck. For example, load a package_1 into a
```

```
nto
ge and truck are

(and
    (not (package-at ?p ?l))
    (package-in-truck ?p ?t)
)
```

Example from Logistics domain

RL Systems can Benefit Significantly with partially correct symbolic models!
[The Kind LLMs are only Too Happy to give!]

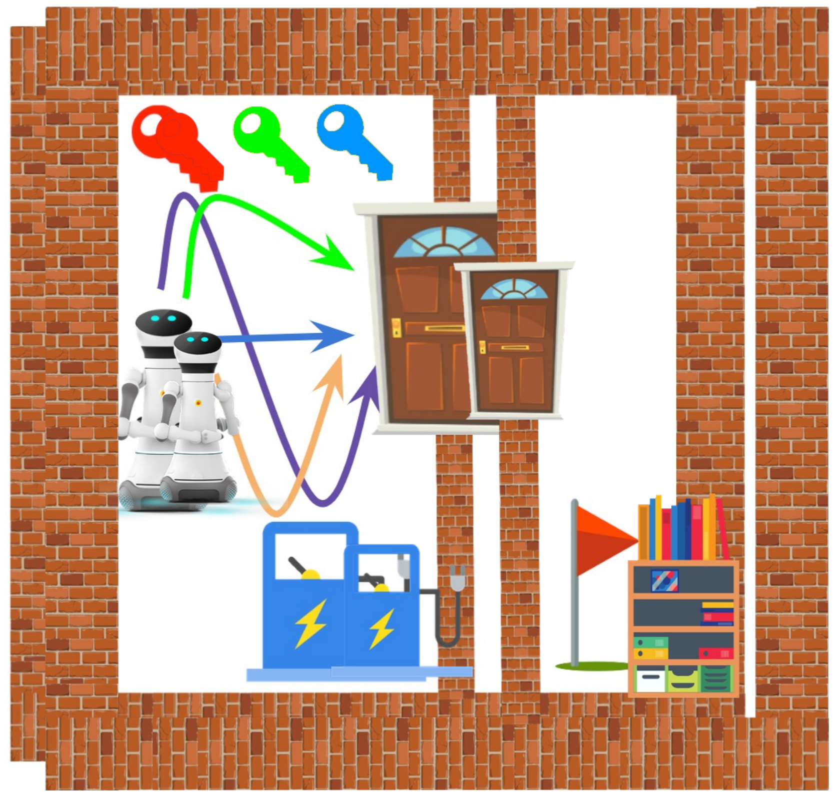
Incomplete Symbolic Model
- Includes potentially **missing information** and **mistakes**
- But still provides useful information about task

Extract information from the model that is guaranteed to be correct

Use landmarks as subgoals
- Example: door-open, at-destination ...

Derive reward functions

Diverse set of skills learned per landmark
- Example: multiple ways to get to the door in the image on the right



[ICML 2022]

Code as (Hierarchical) Policies

Original LLM Prompt

```
import numpy as np
from utils import get_obj_bbox_xyxy
# define function: total = get_total(xs).
def get_total(xs):
    return np.sum(xs)
# define function: get_objs_bigger_than_area_th(obj_names, bbox_area_th).
```

LLM Response with Unrecognized Function

```
def get_objs_bigger_than_area_th(obj_names, bbox_area_th):
    return [name for name in obj_names
            if get_obj_bbox_area(name) > bbox_area_th]
```

Code Parser

Prompts again for any missing functions found

Extracts task reduction schema from LLM!

Re-prompt

```
# define function: get_obj_bbox_area(obj_name).
```

LLM Response

```
def get_obj_bbox_area(obj_name):
    x1, y1, x2, y2 = get_obj_bbox_xyxy(obj_name)
    return (x2 - x1) * (y2 - y1)
```

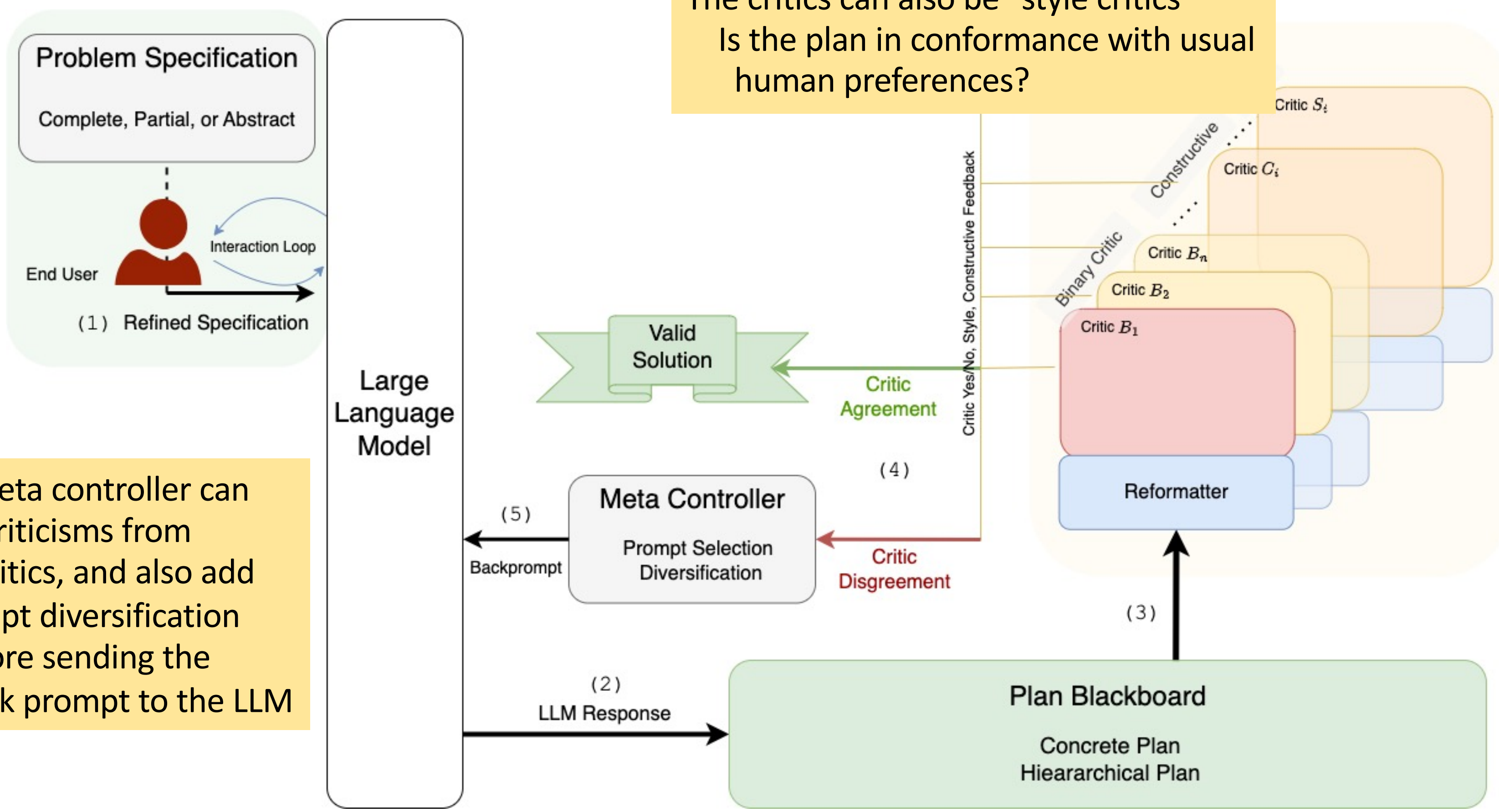
Parses new response and recursively prompts for any missing functions if needed

Liang, Jacky, et al. "Code as policies: Language model programs for embodied control." *arXiv preprint arXiv:2209.07753* (2022). Prompts are from that paper.

LLM-Modulo with a Bank of critics

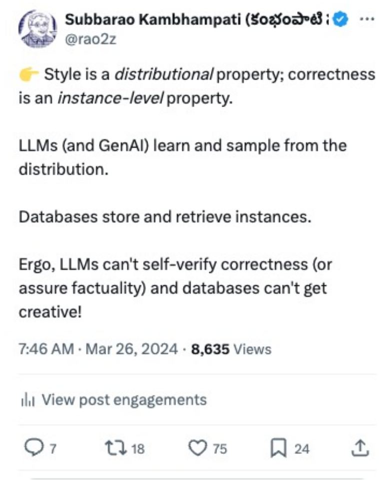
We can accommodate multiple critics
e.g. one to verify causal correctness
one to verify resource usage
Can be constructive critics

The critics can also be "style critics"
Is the plan in conformance with usual human preferences?

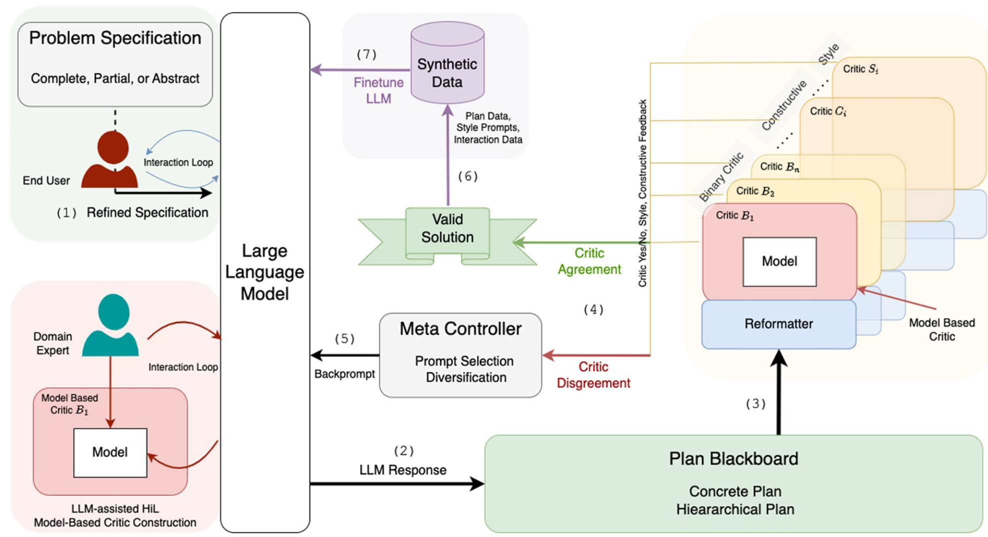


The meta controller can pool criticisms from the critics, and also add prompt diversification before sending the back prompt to the LLM

Types of Critics in LLM-Modulo



- Correctness vs. Style
  - LLMs can't directly critic correctness
    - But can help in obtaining the model driving the critics
  - LLMs can be directly used for critiquing style
- Critics can be
    - Binary ("try again"),
    - Constructive
      - Point out errors in the candidate
      - Suggest local repairs
- Meta controller combines the criticisms from the various critics and sends it as a back prompt
  - Can also do prompt diversification as part of the process



LLMs as Style Critics & Human Preference Proxies

- We investigate the **potential of LLMs to serve as effective human proxies** by capturing human preferences in human-AI collaboration settings.
- LLMs can play different roles in Human-aware AI interaction: as a **Human Proxy**, Translator (common lingua franca), and the Actor.
- **Theory of Mind (ToM)** requires LLMs to also be able to capture human mental states, desires, and beliefs for **reward design/learning mechanisms**.
- Human-aware AI agents can incorporate such reward functions to account for **human-in-the-loop's preferences**.

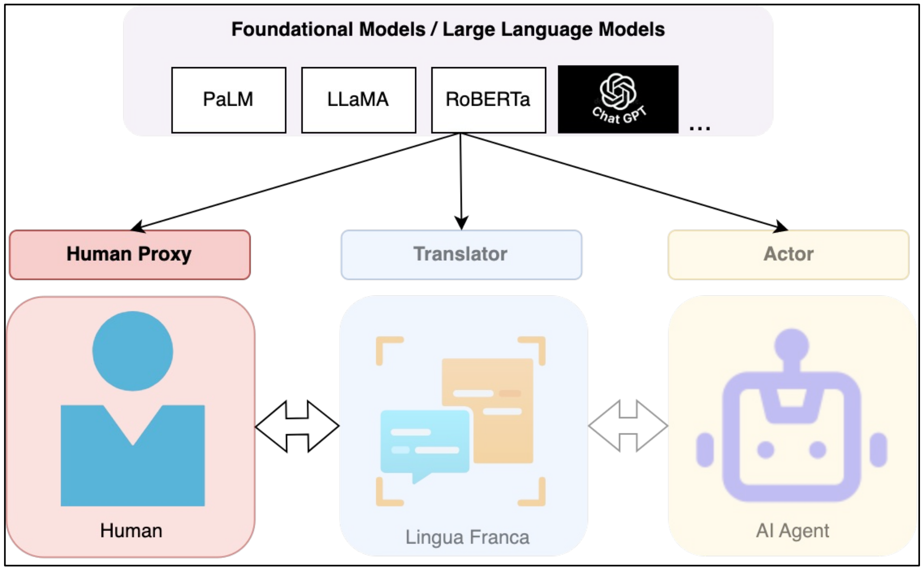


Figure: Different roles of an LLM in Human-AI interaction.

Theory of Mind abilities of Large Language Models in Human-Robot Interaction : An Illusion?
Mudit Verma, Siddhant Bhambri*, Subbarao Kambhampati.*
HRI 2024

LLMs as Human Preference Proxies

Can LLMs capture human preferences?

Probing LLMs with explicability preferences:

- Under explicability preference, the **human expects the agent to behave in a certain way**, and the agent proactively attempts to model this expectation and follow it.
- Here, the human takes the role of an **observer**.

Probing LLMs with sub-task specification preferences:

- We consider a Human-AI teaming scenario where the **human plays an active role and can perform actions** in the world alongside the AI agent.
- **Sub-task specification preferences** involve the agent to produce the same set of sub-tasks that the human has in mind to achieve the team objective.

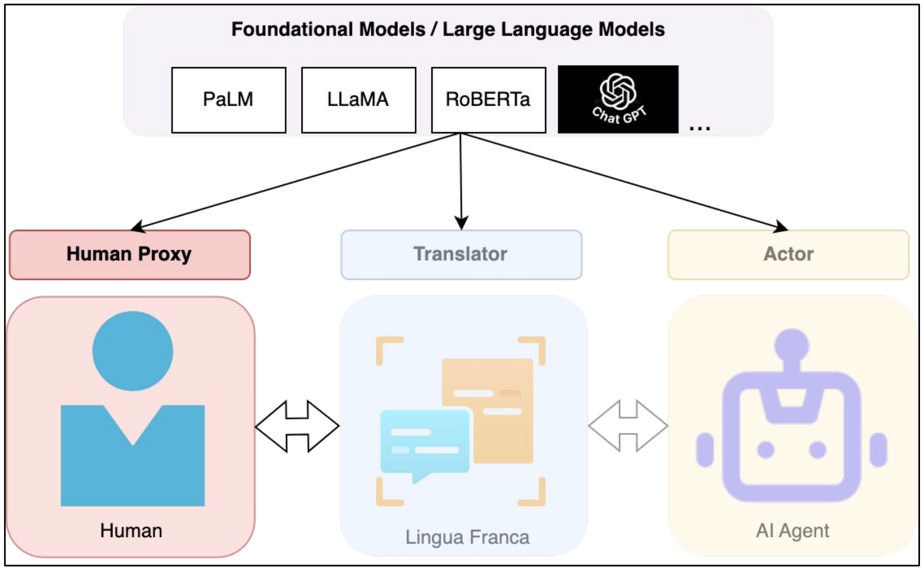


Figure: Different roles of an LLM in Human-AI interaction.

LLMs as Behavior Critics to catch undesirable robot behaviors

Can LLMs capture human preferences in embodied AI tasks?

- It may be intractable to construct formal verifiers for tasks that have a wide scope.
- LLMs or VLMs can be a proxy of common human preferences and undesirability

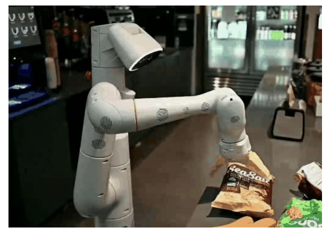
- We evaluated GPT-4V with videos of diverse suboptimal behaviors
- GPT-4V critic catches 69% of undesirable behaviors (recall rate) while only 62% of the critiques are valid (precision rate)

- Results confirm the broadness of GPT-4V's knowledge & the subpar precision of its outputs

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors

Lin Guan, Yifan Zhou*, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati.*

Pick up a bag of chips

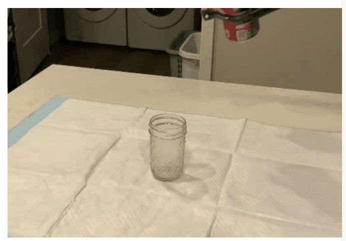


GPT-4V: The robot gripped the bag of chips too tightly, causing it to crumple and potentially damaging the contents.

Place knife on board

GPT-4V: The robot released the knife from a height that caused it to bounce upon hitting the cutting board, which is potentially dangerous due to the sharp blade.

Pour coke into the glass



GPT-4V: The robot poured the coke too quickly, causing excessive fizzing and overflow of the liquid from the glass.

Serve orange juice

GPT-4V: The robot's arm holding the glass of orange juice passes over the laptop, which poses a risk of spilling liquid on the electronic device.

Hand scissors to human

GPT-4V: The scissors are handed over with the pointy ends facing the person, which poses a risk of injury.

Place facial cleanser

GPT-4V: The robot dispensed facial cleanser onto the tray instead of placing the bottle onto the tray.

LLMs as Behavior Critics to catch undesirable robot behaviors

Can LLMs capture human preferences in embodied AI tasks?

- It may be intractable to construct formal verifiers for tasks that have a wide scope.
- LLMs or VLMs can be a proxy of common human preferences and undesirability

- We evaluated GPT-4V with videos of diverse suboptimal behaviors
- GPT-4V critic catches 69% of undesirable behaviors (recall rate) while only 62% of the critiques are valid (precision rate)

- Results confirm the broadness of GPT-4V's knowledge & the subpar precision of its outputs

Place vessel onto burner

GPT-4V:
The robot placed the vessel off-center on the burner, which could lead to uneven heating or potential tipping of the vessel.

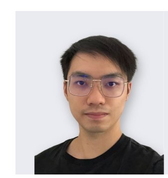
(grounding error) The robot released the vessel from a height that could cause damage to the vessel or the stove if it were heavier or more fragile.

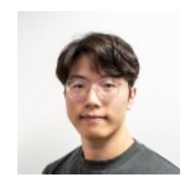
Move spoon to bowl

GPT-4V:
(unoperationalizable) The robot caused pasta sauce to drip onto the counter while transferring the spoon from the jar to the bowl.

Note: GPT-4V correctly detects the sauce spillage but does not provide the fundamental reason for why it occurred, thereby making it difficult to ascertain actionable advice for behavior correction.

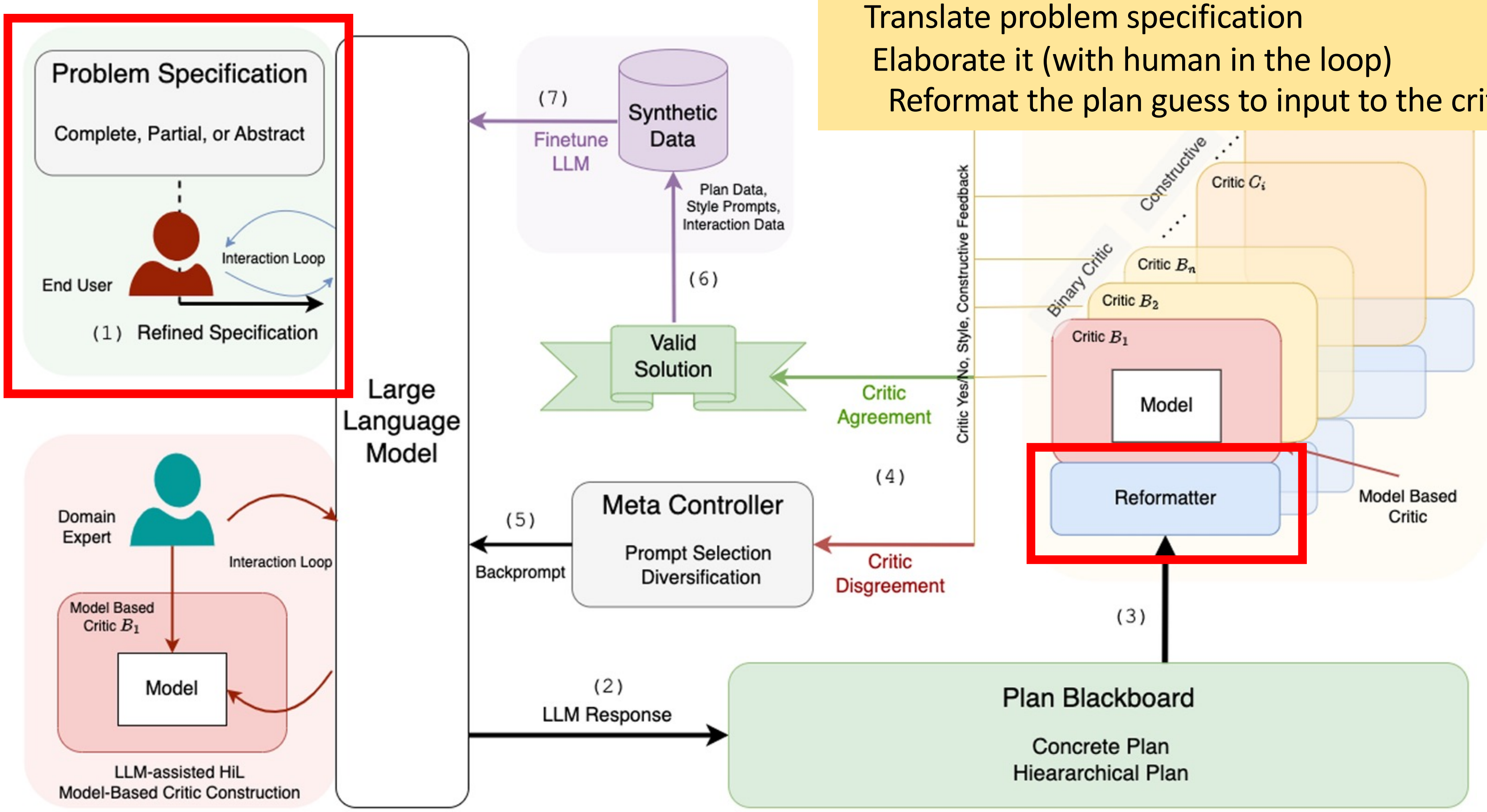
"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors
Lin Guan, Yifan Zhou*, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati.*





LLMs for Format Change/Specification Elaboration

Given that LLMs are good at format change
 they can be used to
  Translate problem specification
   Elaborate it (with human in the loop)
    Reformat the plan guess to input to the critic



LLMs for format change

- We have used them in the past to **translate** existing documents in natural language to formal representations

GPT3-to-Plan: Extracting Plans from Text
[KEPS-21]

TEXT: Windows Help and Support
Start internet explorer.
You need to eventually click on the advanced tab
But before that, click on the internet options on the tools menu.
After the click on the advanced tab, click to clear or select the enable personalized favorites menu check box.
ACTIONS:
click(internet,explorer), click(tools), click(internet,options), click(advanced), click(personalized,favorites), click(check,box)

TEXT: CookingTutorial
For this recipe you need to let the beef simmer for 30 minutes.
After that, make sure that you don't start cooking oats right away.
You have to first measure their quantity which needs to be 150g.
Then cook them and pour the vegetables with the meat.
ACTIONS:
let(beef,simmer), measure(quantity,oats), cook(oats), pour(vegetables,meat)

TEXT: WikiHow Home and Garden
You can decorate your floor anytime.
But before decorating your floor it's important that you paint your walls first.
After painting your walls, remove all furniture in the room.
ACTIONS:
paint(walls), remove(furniture), decorate(floor)

Figure 2: Query examples on WHS, CT and WHG. Each query was input to Davinci along with two preceding training instances containing the largest proportion of *optional* and *exclusive* actions. The output is shown in regular text while the input is displayed in bold.

Text to plan using GPT-3

- ❑ Workshop on KEPS (ICAPS'21)
- ❑ Workshop on Planning for Financial Services (ICAPS'21)

TEXT: Windows Help and Support

Start internet explorer.
You need to eventually click on the advanced tab
But before that, click on the internet options on the tools menu.
After the click on the advanced tab, click to clear or select the enable personalized favorites menu check box.

ACTIONS:
click(internet,explorer), click(tools), click(internet,options), click(advanced), click(personalized,favorites), click(check,box)

TEXT: CookingTutorial

For this recipe you need to let the beef simmer for 30 minutes.
After that, make sure that you don't start cooking oats right away.
You have to first measure their quantity which needs to be 150g.
Then cook them and pour the vegetables with the meat.

ACTIONS:
let(beef,simmer), measure(quantity,oats), cook(oats), pour(vegetables,meat)

TEXT: WikiHow Home and Garden

You can decorate your floor anytime.
But before decorating your floor it's important that you paint your walls first.
After painting your walls, remove all furniture in the room.

ACTIONS:
paint(walls), remove(furniture), decorate(floor)

Figure 2: Query examples on WHS, CT and WHG. Each query was input to Davinci along with two preceding training instances containing the largest proportion of *optional* and *exclusive* actions. The output is shown in regular text while the input is displayed in bold.

	Action names			*Action arguments*		
Model	WHS	CT	WHG	WHS	CT	WHG
EAD	86.25	64.74	53.49	57.71	51.77	37.70
CMLP	83.15	83.00	67.36	47.29	34.14	32.54
BLCC	90.16	80.50	69.46	93.30	**76.33**	70.32
STFC	62.66	67.39	62.75	38.79	43.31	42.75
EASDRL	93.46	84.18	75.40	**95.07**	74.80	75.02
cEASDRL	**97.32**	**89.18**	**82.59**	92.78	75.81	**76.99**
GPT-3						
Davinci	**86.32**	**58.14**	**43.36**	22.90	**29.63**	**22.25**
Curie	75.80	35.57	22.41	**31.75**	22.16	13.79
Babbage	62.59	20.62	14.95	22.91	12.59	7.33
Ada	60.68	14.68	8.90	17.91	4.13	2.27

Table 3: F_1 scores for all actions and their arguments across the WHS, CT and WHG datasets for the state-of-art sequence extraction models and GPT-3. State-of-art task-specific model F_1 scores are extracted from Miglani and Yorke-Smith (2020); Feng, Zhuo, and Kambhampati (2018) and represent their best possible recorded performance.

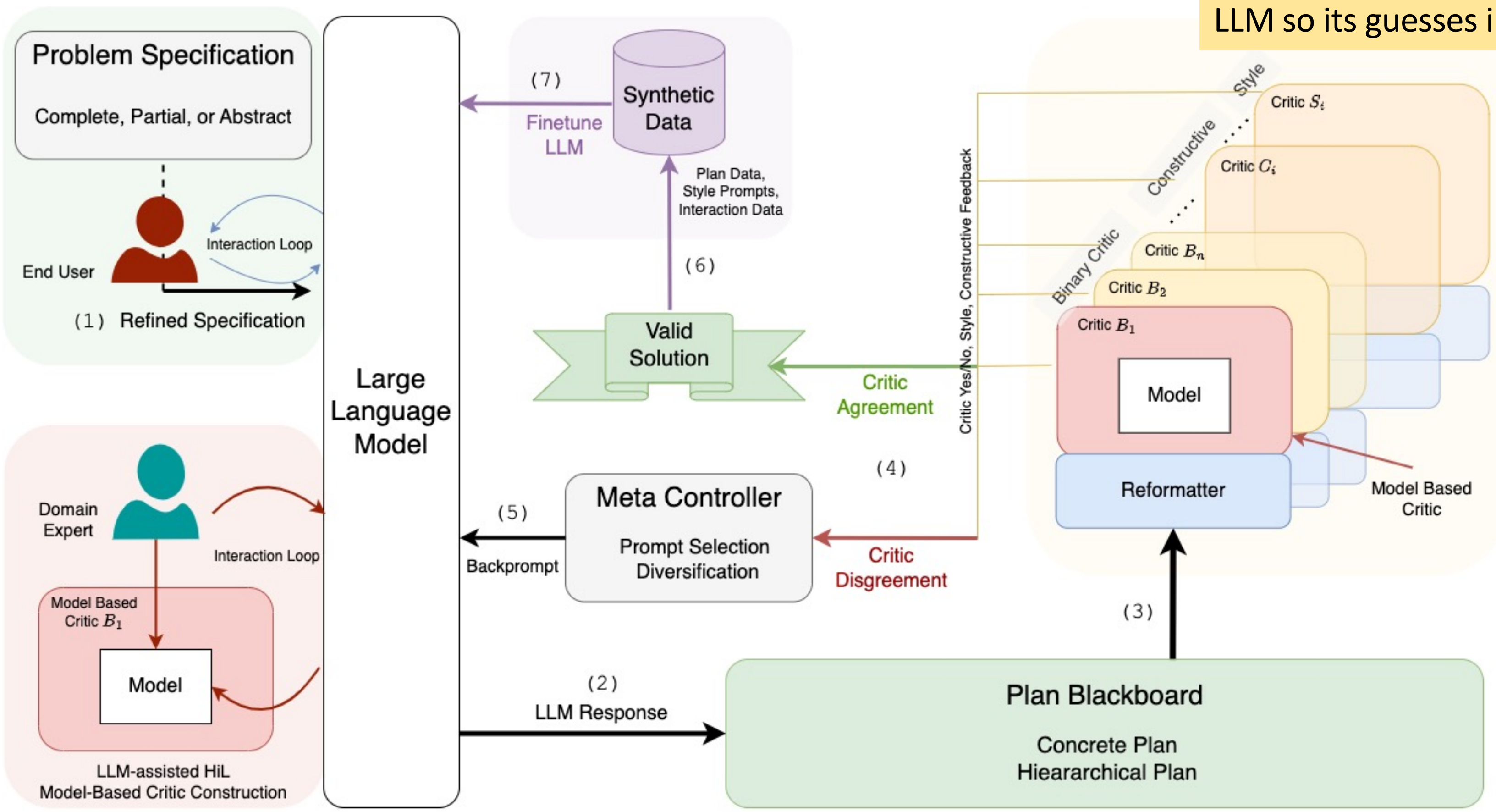
- We investigated how **GPT-3**, one of the most recent transformer-based language models, can be used to extract structured actions from natural language texts. We find that these models achieve comparable, and in some cases better scores than previous state of the art task-specific methods
- **Impact:** Existing knowledge in the form of textual procedures and plans can be translated into formal representations to aid novice Navy personnel understand and carry out complex procedures. The translated procedures can also be leveraged by other automated systems in-place.

Using LLMs to translate Goals specified in Natural Language

- Perhaps the least ambitious way to use LLMs in plan generation is to just have them convert the goals specified in natural language to formal representations (..and then use an actual external planner to solve the planning problem..)
  - A bit of a Rube Goldberg approach..
- Examples (not by us) include
  - Converting goal specifications to PDDL spec (LLM+P)
    - Which basically involves putting in parentheses at the lowest end..
  - Converting goal specification to STL spec (AutoTamp)

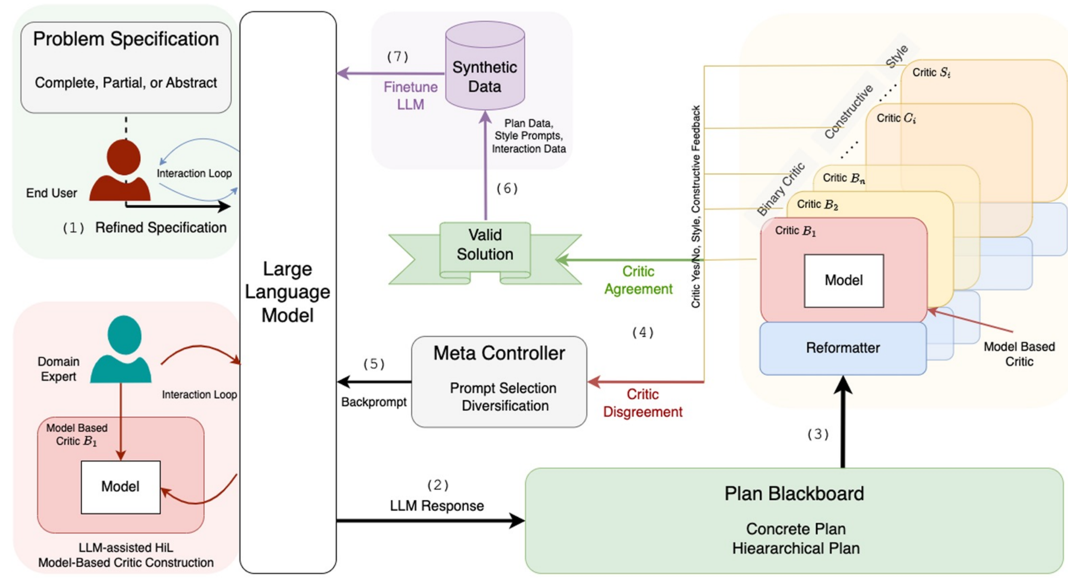
Generating Synthetic Data (Self-Instruct LLM-Modulo Way)

Finally, since the solutions coming out of LLM-Modulo frameworks are sound, they can be used to build a corpus of synthetic data that can be used to fine-tune the LLM so its guesses improve..



LLM-Modulo Framework: Summary

- LLM-Modulo is a generate-test framework with LLMs generating candidate plans and critics testing/critiquing them
- LLMs play a variety of constructive roles
  - Generate candidate plans
  - Be an approximate source of models driving the correctness critics
  - Act as style critics
  - Help collating the criticisms from critics (and diversify the prompts as needed)
  - Help with format change—specification level, converting to critic representations
- Preference for critics over solvers
  - Correctness vs. Style
  - Binary vs. Critical feedback vs. Constructive critics
- Human intervention is minimized
  - Once per domain: Teasing out domain model
  - Once per problem: Specification elaboration
  - Humans are not required to be in the inner loop of the back-prompting search



Related work: FunSearch, Alpha Geometry

Also related to the "Compound AI Systems" movement

Is LLM-Modulo just Shoe-Horning LLMs?

(Why bother with LLMs when we already have formal planning systems?)

- Formal planning systems provide soundness and completeness guarantees
  - ..but only with respect to the class of problems they can handle
    - ..for which there are hand-coded/learned models
  - It becomes the end user's responsibility to check if their problem falls in the class handled by a planning system!
- In contrast, LLMs will always guess solutions-–albeit without guarantees
- LLM-Modulo framework is an attempt to keep the best of both worlds
  - Allow end user to pose any problem;
    - Ensure that the solution being sent out is verified by the bank of critics..

ELSEVIER

Artificial Intelligence
Volume 48, Issue 3, April 1991, Pages 261-297

Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services

Jon Doyle, Ramesh S. Patil

Show more

Add to Mendeley Share Cite

https://doi.org/10.1016/0004-3702(91)90029-J

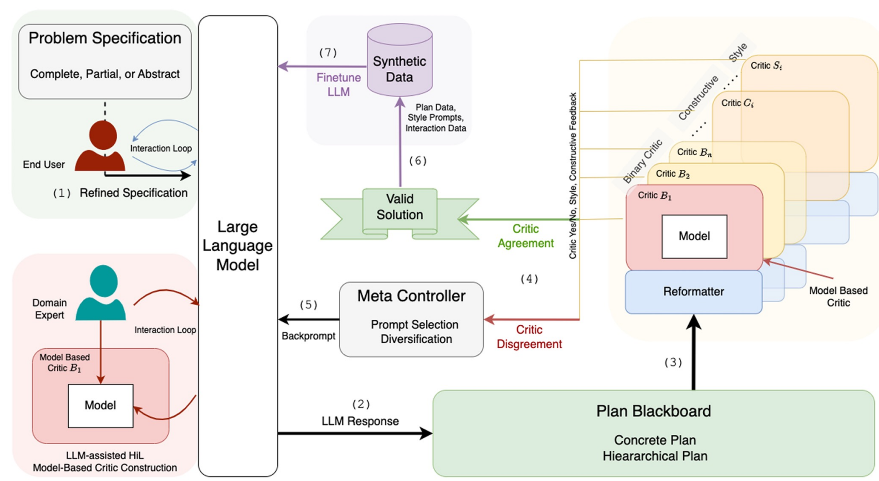
Get rights and content

Abstract

Levesque and Brachman argue that in order to provide timely and correct responses in the most critical applications, general-purpose knowledge representation systems should restrict their languages by omitting constructs which require nonpolynomial worst-case response times for sound and complete classification. They also separate terminological and assertional knowledge, and restrict classification to purely terminological information. We demonstrate that restricting the terminological language and classifier in these ways limits these "general-purpose" facilities so severely that they are no longer generally applicable. We argue that logical soundness, completeness, and worst-case complexity are inadequate measures for evaluating the utility of representation services, and that this evaluation should employ the broader notions of utility and rationality found in decision theory. We suggest that general-purpose representation services should provide fully expressive languages, classification over relevant contingent information, "approximate" forms of classification involving defaults, and rational management of inference tools.

Talk Overview

- Part 1
  - Evaluating LLM Planning capabilities in Autonomous mode, including effect of
    - Prompting strategies (including Chain-of-thought)
    - Fine Tuning
    - Self Verification
  - Understanding the contradictory claims in the literature
- Part 2
  - Sane roles of LLMs in Planning (with LLM-Modulo frameworks)
    - LLMs as heuristics, LLMs as candidate generators
    - Back prompting from external verifiers
    - LLMs as sources of domain models (with humans in the loop)
    - LLMs as format changers/specification elaborators1
- **Part 3**
  - Some thoughts on why people tend to ascribe planning/reasoning abilities to LLMs



LLM's Approximate Retrieval upends our intuitions re: their guesses

Computational Complexity of the underlying task has no bearing on LLM guesses

- The underlying complexity of the problem has no impact on the LLM's ability to *guess* the answer
  - They are just as fast in guessing answers to undecidable questions as they are in guessing answers to constant time questions
    - ..and in neither case do they have any guarantees about their guess
- Corollary: The usual problem characteristic—*Stochasticity, Partial Observability etc.* — that make it computationally harder *don't matter* in LLM's ability to guess
- After all, they take constant time per token
  - ..and no, asking LLMs to "pause" doesn't change any of this!

Background Knowledge is *easier* for LLMs (approximately..)

- Much has been made in traditional AI of the difficulty of getting relevant knowledge.
- Having been trained on web-scale collective knowledge of humanity, LLMs are remarkably better at this
- They are pretty good (with no guarantees—and some brittleness) at
  - Commonsense
  - Domain knowledge
  - Theory of Mind
  - Analogies
- (In addition, of course, to linguistic abilities such as summarization, elaboration, format change etc.)

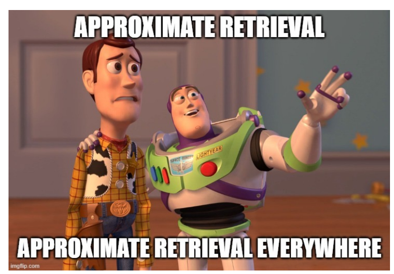
Why are LLMs claimed to do Reasoning/Planning?

Approximate omniscience of LLMs allows them to fake reasoning by retrieval

- Memory reduces the need to reason from first principles.
  - "Why are manhole covers round?"
- The training corpus is the entire web, and it is hard for anyone to know what it already contained
- The web corpus contains both base facts and *deductive closure facts*
  - Retrieval of the later can be mistaken for reasoning
- Fine tuning and training from synthetic data further muddy waters by deliberately converting reasoning into approximate retrieval
  - Think compiling *someone's system 2 to your system 1*

LLMs may approximate reasoning with pattern finding

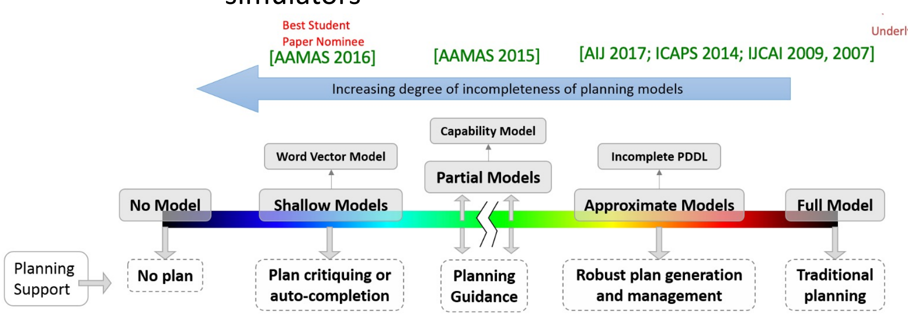
- Think of trying to predict the satisfiability of a random 3-SAT instance
- Suppose you train a learner with a gazillion random 3-SAT instances
- Will it discover Davis-Putnam procedure or is it more likely to discover the sharp phase transition?
  - Easier to find latent variables corresponding to #clauses/#variables, and learn a rule to classify instances that way



Planning in the age of LLMs

For far too long, there has been a race to bottom on the level of knowledge given to planners

- Planning started knowledge-based
  - Remember, Noah was an HTN planner, y'all!
  - ..and fell to ground propositional level –because it seemed too unseemly to depend on humans for these knowledge-based models
    - And focus on doing interaction resolution from first principles
- RL was worse—propositional was too high-level a knowledge to ask from humans
  - They wanted to say they will learn it all
    - And not have humans give any knowledge about the domain. They just wanted "SIMULATORS",
  - ..and it took for ever to do anything—even with simulators



LLMs change that—rather drastically!

- LLM makes it easy to get knowledge without making it look like we are inconveniencing any specific human
  - We are just stealing everything humans told each other—is all.
- ..as long as you relax the requirement of the knowledge actually being "correct"
  - ..then again, do you really believe that huge human-written models are correct?
- So the million dollar qn is: How would you do planning if you have some doddering know-it-all ready to give you any kind of knowledge
  - "Actions and effects"
  - "Task reduction schemas"
  - "Cases"
- Time for LLM-HTN, LLM-CBR etc. paradigms
  - **Or even a resurrection of the model-lite planning dream..**

Epilogue

LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

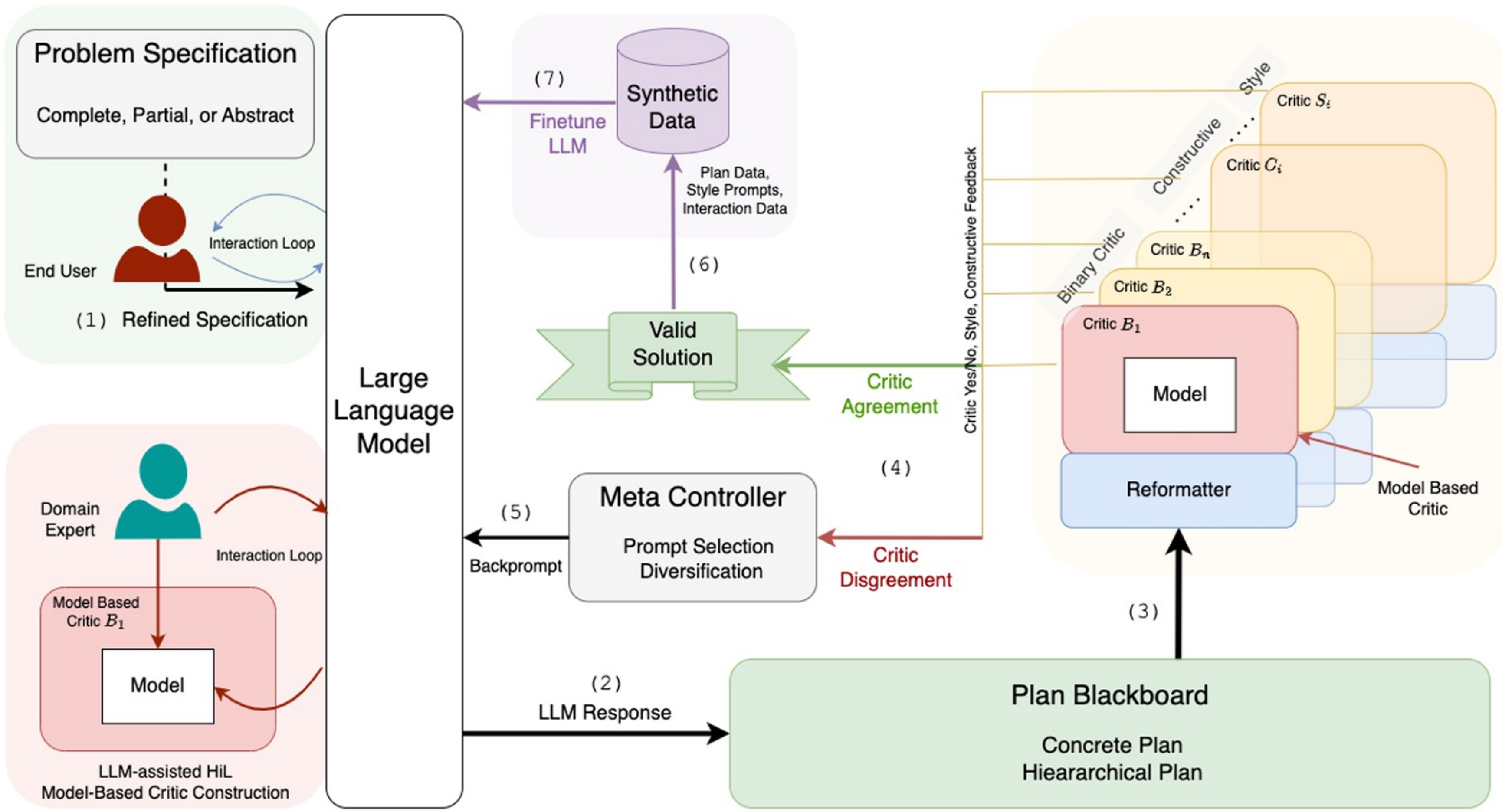
LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)
  - In the LLM-Modulo framework, LLMs can play multiple roles
    - Guess plans
    - Guess domain models
    - Help elaborate the problem specification
    - Translate formats

LLM-Modulo Frameworks for Planning



Claims on LLMs Reasoning/Planning Abilities

Over-optimism

- LLMs can reason/plan
- With just the right "prompt"
- With letting them verify and critique their solutions

Our Position

- LLMs can't reason/plan
- But they are approximate knowledge sources
- They can play much more meaningful roles in ***LLM-Modulo Settings***

Over-pessimism

- LLMs can't reason/plan
- They can be translators at best
- Let external symbolic solvers handle the problem

What Planning is & What LLMs are good at..

Planning (as used in common parlance) involves

- Planning knowledge
  - Actions, preconditions and effects
  - General Recipes: Task reduction schemata (e.g. HTN planning)
  - Old examples: Case libraries
- Plan generation/verification techniques
  - Interaction analysis/resolution
  - Plan merging techniques
  - Plan modification techniques

LLMs accept any planning problem—even if it not expressible in PDDL standard—and they don't give any correctness guarantees.

AI Planners will give formal guarantees, but only accept problems expressible in their language.

Contrasting what AI Planning & LLMs bring to the table

- AI Planning (aka ICAPS planning) assumes that the planning knowledge is given up front, and focuses generation and verification techniques
  - Emphasis on guaranteeing completeness/correctness of the plans w.r.t. the model
    - By and large the common paradigm—although there have been occasional mutinies
      - Model-Lite Planning approaches
- LLMs, trained as they are on everything ever put on the web, have a kind of "*approximate omniscience*". This helps them spit out actions, recipes, or cases
  - But they lack the ability to stitch the recipes together to ensure that there is no actually interaction free!

Then how come LLMs are trumpeted as doing planning?

- Most cases where LLMs are claimed to generate executable plans, on closer examination, turn out to be cases where LLMs are getting by with the generate approximate recipes step
  - Generate approximate recipes/cases (for common sense domains)
    - e.g. wedding plans
  - Convert tasks into (approximate) task reduction schemas
    - Perhaps written out as "programs" (e.g. Code as Policies..)
      - (SHOP2 schemas were already pseudo lisp code—if only written by humans)
  - LLM-HTN and LLM-CBR differ from HTN and CBR in that they generate the task-reduction schemas or the cases on demand

- And the interaction resolution/search part is
  - either *pushed under the rug*
    - Consider "high level" plans like "wedding plans" for which there are enough generic recipes available in the training set, and are described at a sufficiently high level of abstraction, the execution issues are left to the imagination of the user
  - or *has been pawed off to human prompters who are required to give "hints" to the LLM to come up with plan variants that are (more) correct*
    - Note that here the human is essentially playing the role of an external verifier & critic
      - In cases where the humans are end users not well versed with all details of the domain, they can be faulty verifiers

NEURAL INFORMATION PROCESSING SYSTEMS

Getting Started | Schedule | Tutorials | Main Conference | Workshops | Community | Help | Subbarao

Browse

mini | compact | detail

kambhampati | × | title | author | session

shuffle by serendipity

showing 3 of 3 papers

On the Planning Abilities of Large Language Models - A Critical Investigation

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, Subbarao Kambhampati

Tu, Dec 12, 09:45 -- Poster Session 1

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

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, Subbarao Kambhampati

Tu, Dec 12, 16:15 -- Poster Session 2

PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, Subbarao Kambhampati

We, Dec 13, 09:45 -- Poster Session 3

arXiv > cs > arXiv:2402.08115

Computer Science > Artificial Intelligence

[Submitted on 12 Feb 2024]

On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks

Kaya Stechly, Karthik Valmeekam, Subbarao Kambhampati

There has been considerable divergence of opinion on the reasoning abilities of Large Language Models (LLMs). While the initial optimism that reasoning might emerge automatically with scale has been tempered thanks to a slew of counterexamples--ranging from multiplication to simple planning--there persists a wide spread belief that LLMs can self-critique and improve their own solutions in an iterative fashion. This belief seemingly rests on the assumption that verification of correctness should be easier than generation--a rather classical argument from computational complexity--which should be irrelevant to LLMs to the extent that what they are doing is approximate retrieval. In this paper, we set out to systematically investigate the effectiveness of iterative prompting in the context of reasoning and planning. We present a principled empirical study of the performance of GPT-4 in three domains: Game of 24, Graph Coloring, and STRIPS planning. We experiment both with the model critiquing its own answers and with an external correct reasoner verifying proposed solutions. In each case, we analyze whether the content of criticisms actually affects bottom line performance,

arXiv > cs > arXiv:2402.04210

Computer Science > Artificial Intelligence

[Submitted on 6 Feb 2024]

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors

Lin Guan, Yifan Zhou, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati

Large-scale generative models are shown to be useful for sampling meaningful candidate solutions, yet they often overlook task constraints and user preferences. Their full power is better harnessed when the models are coupled with external verifiers and the final solutions are derived iteratively or progressively according to the verification feedback. In the context of embodied AI, verification often solely involves assessing whether goal conditions specified in the instructions have been met. Nonetheless, for these agents to be seamlessly

Check for updates

DOI: 10.1111/nyas.15125

ANNALS OF THE NEW YORK ACADEMY OF SCIENCES

COMMENTARY

THE SCIENCES

Can large language models reason and plan?

Subbarao Kambhampati

School of Computing & Augmented Intelligence, Arizona State University

Correspondence

Email: rao@asu.edu

Abstract

While humans sometimes do show the capability of correcting their own erroneous guesses with self-critiquing, there seems to be no basis for that assumption in the case of LLMs.

Large language models (LLMs), essentially *n*-gram models on steroids[a] that have been trained on web-scale language corpora (or, effectively, our civilizational knowledge), have caught our collective imagination with linguistic behaviors that no one expected text completion systems to possess.[1] By training and operation, LLMs are perhaps best seen as giant non-veridical memories akin to an external System 1 (Ref. 2) for us all (see Figure 1). Their seeming versatility has however led many researchers to wonder whether they can also do well on planning and reasoning tasks typically associated with System 2 competency.

Nothing in the training and use of LLMs would seem to suggest remotely that they can do any type of principled reasoning (which, as we know, often involves computationally hard inference/search). What national Planning Competition (IPC)—including the well-known Blocks World[c]. Our results[4] were contrary to the anecdotal claims about the planning abilities of LLMs, and when we made them public, received significant attention in the AI circles.

By the beginning of 2023, with the wide-spread public release of ChatGPT, and later, GPT4, there were a slew of additional claims, including in refereed papers, about LLM's abilities to reason and plan. So we decided to repeat our tests on both GPT3.5 and GPT4.[5] Initial results showed that there was some improvement in the accuracy of generated plans from GPT3 to GPT3.5 to GPT4, with GPT4 reaching 30% empirical accuracy in the Blocks World (albeit still lower in other domains). We then wanted to know whether the modest improvement

arXiv > cs > arXiv:2402.01817

Computer Science > Artificial Intelligence

[Submitted on 2 Feb 2024 (v1), last revised 6 Feb 2024 (this version, v2)]

LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, Anil Murthy

There is considerable confusion about the role of Large Language Models (LLMs) in planning and reasoning tasks. On one side are over-optimistic claims that LLMs can indeed do these tasks with just the right prompting or self-verification strategies. On the other side are perhaps over-pessimistic claims that all that LLMs are good for in planning/reasoning tasks are as mere translators of the problem specification from one syntactic format to another, and ship the problem off to external symbolic solvers. In this position paper, we take the view that both these extremes are misguided. We argue that auto-regressive LLMs cannot, by themselves, do planning or self-verification (which is after all a form of reasoning), and shed some light on the reasons for misunderstandings in the literature. We will also argue that LLMs should be viewed as universal approximate knowledge sources that have much more meaningful roles to play in planning/reasoning tasks beyond simple front-end/back-end format translators. We present a vision of {\bf LLM-Modulo Frameworks} that combine the strengths of LLMs with external model-based verifiers in a tighter bi-directional interaction regime. We will show how the models driving the external verifiers themselves can be acquired with the help of LLMs. We will also argue that rather than simply pipelining LLMs and symbolic components, this LLM-Modulo Framework provides a better neuro-symbolic approach that offers tighter integration between LLMs and symbolic components, and allows extending the scope of model-based planning/reasoning regimes towards more flexible knowledge, problem and preference specifications.

Subjects: **Artificial Intelligence (cs.AI)**; Machine Learning (cs.LG)

Cite as: arXiv:2402.01817 **[cs.AI]**

(or arXiv:2402.01817v2 **[cs.AI]** for this version)

arXiv > cs > arXiv:2401.05302

Computer Science > Robotics

[Submitted on 10 Jan 2024 (v1), last revised 17 Jan 2024 (this version, v2)]

Theory of Mind abilities of Large Language Models in Human-Robot Interaction : An Illusion?

Mudit Verma, Siddhant Bhambri, Subbarao Kambhampati

Large Language Models have shown exceptional generative abilities in various natural language and generation tasks. However, possible anthropomorphization and leniency towards failure cases have propelled discussions on emergent abilities of Large Language Models especially on Theory of Mind (ToM) abilities in Large Language Models. While several false-belief tests exists to verify the ability to infer and maintain mental models of another entity, we study a special application of ToM abilities that has higher stakes and possibly irreversible consequences : Human Robot Interaction. In this work, we explore the task of Perceived Behavior Recognition, where a robot employs a Large Language Model (LLM) to assess the robot's generated behavior in a manner similar to human observer. We focus on four behavior types, namely - explicable, legible, predictable, and obfuscatory behavior which have been extensively used to synthesize interpretable robot behaviors. The LLMs goal is, therefore to be a human proxy to the agent, and to answer how a certain agent behavior would be perceived by the human in the loop, for example "Given a robot's behavior X, would the human observer find it explicable?". We conduct a human subject study to verify that the users are able to correctly answer such a question in the curated situations (robot setting and plan) across five domains. A first analysis of the belief test yields extremely positive results inflating ones expectations of LLMs possessing ToM abilities. We then propose and perform a suite of perturbation tests which breaks this illusion, i.e. Inconsistent Belief, Uninformative Context and Conviction Test. We conclude that, the high score of LLMs on vanilla prompts showcases its potential use in HRI settings, however to possess ToM demands invariance to trivial or irrelevant perturbations in the context which LLMs lack.