Neither LLMs nor LRMs have the ability to go beyond the humanity's knowledge closure

A note by Subbarao Kambhampati, July 16, 2025

Neither LLMs nor LRMs have the ability to go beyond the humanity's knowledge closure--which is needed for true discoveries. Both are beholden to the collected knowledge of the humanity (whether declaratively enumerated or procedurally specified as models/"verifiers").

*There is little doubt that LLMs--trained as they are on the collective digital footprints of humanity--"know" more than any single human.*

*The AGI-level question is whether they know more than the humanity.*

*Unlike search engines, LLMs are more than just a passive repository of human knowledge--they do have the ability to combine pieces. This "information integration" is their biggest use case right now (modulo the hallucination issue 2).*

See Appendix 1.

The newfound love for verifiers--which are written by humans after all--is but a way for LRMs to train themselves over synthetic data--Generate candidate solutions--have 'em checked by the verifiers. Rinse, then repeat, with some LLM-Modulo + RL combo (c.f. [3])

Just as RL over simulators (written by humans) is a lazy way to compile simulator knowledge into the policy; LLM-RL over verifiers too is a form of compiling (human written) verifier knowledge (into the base LLM) (c.f Appendix 2)

LLMs/LRMs are great as force multipliers. But if you really want your AI agent to learn and go beyond the humanity's knowledge closure, you also need your agent to act in the world (**not** simulator) and learn from that (c.f. Yann LeCun's exhortations).

Acting in the world--as against learning vicariously from other's experiences--is fraught--especially given that much of real world is non-ergodic. It is okay for LLMs/LRMs to skip this part--as long as we avoid the "automated scientific discoveries by next weekend" hype.

[Added] This is not to say that finding everything in humanity's knowledge closure is trivial! Most math is indeed in the deductive closure of the axioms and no wonder LRMs continue to have impressive success there (modulo the curmudgeon Gödel and his eventual incompleteness ).

[Added] Some confuse the fact that LLMs can generate (hallucinate) all sorts of ideas as an indication that they can do discoveries. No. They can \*help\* humans do discoveries. Idea/hypothesis generation is only one part of Science! (c.f. [4]).

Test-time-scaling, Post-training and Distillation are just compiling the verifier signal into the LLM at different phases #SundayHarangue

Most documented advances of LRMs on reasoning problems have been on tasks for which there are formal verifiers from traditional AI and Computer Science. The modus operandi of current LRMs is leveraging these verifiers in a generate-test loop at test time, training time or distillation time in order to partially compile the verification signal into generation.

In other words, post-training LRMs can be seen as iteratively compiling reasoning into (approximate) retrieval. This iteration is needed because, for reasoning problems which can be arbitrarily scaled in complexity (e.g. multi-digit multiplication with increasing digit numbers), an LLM trained on instances of a certain size quickly loses its ability to provide good guesses at larger sizes (see Appendix 3) .

Post-training approaches depend on the ability of the base LLM to have high enough top-k accuracy (i.e., be capable of generating at least one correct solution given k guesses) so that the verifier has something to select (otherwise, there is no signal either for fine tuning or the RL phase!).

This general idea is consistent with the dictum (attributed to Marvin Minsky) that intelligence is shifting the test part of generate-test into the generate part. In particular, using verifiers at test time has already been advocated by the LLM-Modulo framework.

Indeed, LRM post-training approaches crucially depend on the signal from the verifier to separate trajectories supplied by the base LLM into those that reach correct solutions vs. those that don’t (and thus, this can be seen as a form of “train time LLM-Modulo”).

Once this is done, these traces are used to refine the base LLM (“generator”) via either fine tuning or RL. This part can thus be interpreted as partially compiling the verifier signal into the generator. Finally, while Deepseek R1 just deploys the refined LLM at inference stage, without resorting to any test time verification, they do wind up using verifiers when they develop additional synthetic data with the help of R1 to distill other models.

One way of seeing this training-, test-, and distillation-time verification is as a staged approach to compile the verification signal into an underlying LLM.

In particular, the base LLM used for R1 already has the capability of generating plausible solution trajectories (potentially from the derivational trace data that was already present in the pre-training data). Post-training can be seen as further refining it to come up with accurate solutions for longer/harder problems in fewer tries. Distillation can be seen as propagating this even further.

At each stage, the verification signal is being compiled into the underlying LLM for longer and longer “inference horizons.”

This understanding is consistent with studies on the effectiveness of Chain of Thought ([5]), use of internal vs. external planning approaches for games ([6]), as well as self-improvement in transformers ([7]). In the last case, we would qualify any “self-improvement” claims by saying that it is more the case of incrementally compiling the verifier signal into the base LLM.

# References

[1] [Original note by Subbarao Kambhampati, Linkedin, July 16, 2025](https://www.linkedin.com/posts/subbarao-kambhampati-3260708_neither-llms-nor-lrms-have-the-ability-to-activity-7351769291123314688-aCUo?utm_source=share&utm_medium=member_desktop&rcm=ACoAAAFZfUoBgPoGUucdnvtwuzPv79P8VHj6uvk)

[2] [Reply of Yann LeCun to the original note by Subbarao Kambhampati on Linkedin, July 17, 2025](https://www.linkedin.com/posts/yann-lecun_neither-llms-nor-lrms-have-the-ability-to-activity-7351942020648554496-_M7a?utm_source=share&utm_medium=member_desktop&rcm=ACoAAAFZfUoBgPoGUucdnvtwuzPv79P8VHj6uvk)

[3] [(How) Do LLMs Reason/Plan? talk given at Microsoft Research, youtube video, Subbarao Kambhampati, 4/11/25](https://youtu.be/0u2hdSpNS2o?si=3EdV7NEgrZvyuqMM)

[4] [AI for Scientific Discovery [Briefing & Panel Remarks at National Academies workshop on ],   
Subbarao Kambhampati, youtube video, 2024](https://youtu.be/TOIKa_gKycE?si=pUsnypBHUjxYWKWO)

[5] [Can Large Language Models Reason and Plan? Subbarao Kambhampati, 2024](https://github.com/dimitarpg13/large_language_models/blob/main/articles/human_like_reasoning/Can_Large_Language_Models_Reason_and_Plan_Kambhampati_2024.pdf)

[6] [Chain of Thoughtlessness? An Analysis of CoT in Planning, Kaya Stechly, Karthik Valmeekam, 2025](https://github.com/dimitarpg13/large_language_models/blob/main/articles/CoT/Chain_of_Thoughtlessness_An_Analysis_of_CoT_in_Planning_Stechly_2025.pdf)

[7] [Mastering Board Games by External and Internal Planning with Language Models, John Schultz et al, 2025](https://github.com/dimitarpg13/large_language_models/blob/main/articles/planning_and_reasoning/Mastering_Board_Games_by_External_and_Internal_Planning_with_Language_Models_Schultz_2025.pdf)

[8] [Self-Improving Transformers Overcome Easy-to-Hard and Length Generalization Challenges, Nayoung Lee et al, UoWM, 2025](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/self_improvement/Self-Improving_Transformers_Overcome_Easy-to-Hard_and_Length_Generalization_Challenges_Lee_2025.pdf)

# Appendix

## Appendix 1: Musings on the term Approximate Retrieval

Pinning down Approximate Retrieval in LLMs (and, in the process making sense of that NYT law suit)

The term approximate retrieval (that, afaik, I coined to provide a qualitative understanding of what LLMs do c.f. https://cacm.acm.org/blogs/blog-cacm/276268-can-llms-really-reason-and-plan/fulltext), has caught on a bit.

I will write down what I was trying to capture with the term--both because someone asked for a definition, and because it actually has some bearing on that NYT lawsuit!

0. The "approximate" here is about whether the "retrieved" text is an unaltered copy of something that was stored (and not about whether the retrieval key is matched approximately)

1. Given that, underneath it all, LLMs are trained to be n-gram models (if only on steroids--aka ultra-large n), it should be rather non-controversial to say that they cannot guarantee exact retrieval. They are just compact models of

In other words, with the n-gram model, the prompt is working as a "key" into the CPT rather than a key into any stored database. It is used to sample next token iteratively from the learned CPTs (with the context for the (n+1)th token affected by the specific sample selected as the -th token!)

1.1 LLMs are not databases--and are not indexing and retrieving exactly matching records without altering them. The closest analogy to index is context and that is changing. There is certainly no stored record being retrieved.

1.2 LLMs are also not IR engines which, while doing similarity search (i.e., allowing approximate match with the key), still guarantee that what they give out is what was stored (IR doesn't make documents--it just retrieves documents that are similar to the query!). [Another way to see this is that if LLMs were just doing IR, then the ChatGPT essays can be caught by the old *turnitin*-style plagiarism detectors.]

1.2.1. The whole RAG rage can be understood as adding an external IR component to LLMs, where the prompt is used as an actual IR query on an external vector DB, and the stuff retrieved is added back into the prompt (hoping that LLM will summarize it..).

2. It is precisely this neither-DB-nor-IR nature of the n-gram model that gives LLMs their flexibility--of essentially capturing the distribution (manifold) of the text in the corpus (humorously illustrated by the tooth paste tube 👇metaphor that I had seen somewhere )

3. Because of the way n-gram models work, there is never any 100% guarantee that some stored record (be it a program or an NY Times article) is retrieved unaltered. So why is NYT suing OpenAI?

3.1 However, with a long enough context window, and the network capacity, something close to memorization (aka "plagiarization") of long passages is very much possible (as is being shown in that NYT law suit!).

3.2. Interestingly generative ML systems effectively memorizing full passages/images has been observed in other generative models too--and can be interpreted as a failure to learn the distribution. See for example the old study by Sanjeev Arora et. al. on whether GANs really learn the distribution/manifold or memorize parts of it.

[Do GANs actually learn the distribution? An empirical study, Sanjeev Arora, Yi Zhang, 2017](https://arxiv.org/abs/1706.08224)

4. Commercial LLM makers (will) try to play both ends of the approximate retrieval to their advantage.

4.1. When they try to argue NYT law suit, they will no doubt push on the fact that LLMs don't do exact retrieval and so there is no copyright infringement.

4.2 When they push LLMs for "search", they will try instead to bank on the memorization capabilities!

The truth is that there is no 100% way to guarantee or stop either behavior!

If LLM makers try to reduce memorization, they will certainly see that the LLM's ability to masquerade as search engines--already quite questionable will degrade even further

## Appendix 2: On the Fallacy of Reinforcement Learning

Yann LeCun:

Why do so many people insist on calling Reinforcement Learning what is merely zeroth-order / gradient-free / black-box optimization?

And why do so many people insist on applying Reinforcement Learning to what are essentially optimal control or planning problems?

Subbarao Kambhampati

Every once in a while, I read an RL paper and wish they knew about the significant connections between what they are saying, and what is already known in the planning literature.

May be someone should do a tutorial on Planning lessons for #RL.

RL has often been used as a lazy planning method for problems with known but "intractable" models or simulators.

The only "pure" use case for #RL is one where the agent gets rewards from the real world (even if it is non-ergodic and the agent can die!)

A good way to understand PDDL Planning vs. simulator-based RL connections is by analogy to #SAT/#CSP

SAT/CSP have no real knowledge-level differences; they mainly differ in declarativeness & specification flexibility.

Same for PDDL planning vs. simulator-based RL!

Using simulators, but claiming credit for organisms learning from the real world raw in tooth and claw, may well be one of the less readily acknowledged "bitter truths" about #RL

A diagram of a factory and a diagram of a factory

AI-generated content may be incorrect.An Image from Leslie Kaelbling's talk at #ICLR2020. Her contrasting of the simulator-based-RL vs. Planning approaches is of pedagogical value to post-AlexNet AI folks.

## Appendix 3: On the "Chain of Thought" Delusions

(with Karthik Valmeekam and Kaya Stechly)

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 follow procedures and unroll them for the current problem at hand. (After all, if they can do that, they should be able to do verification and general reasoning too--and we know they suck at those).

And yet the practitioners of CoT swear that any and every problem can be solved with LLM by giving it a bit of a CoT help.

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 can be.

The more general the knowledge, the easier it is for the humans to provide it, and higher the degree of reasoning LLM has to do to operationalize it.

Consider STRIPS planning problems. At the most general level, the CoT prompt will just tell the LLM how to get itself to construct a solution that will be provably correct--using progression, regression or causal-link plan correctness. Would LLMs be able to follow this? We checked this using state tracking prompts and showed they have no effect on LLM performance! (The first graphic below👇)

A screenshot of a document

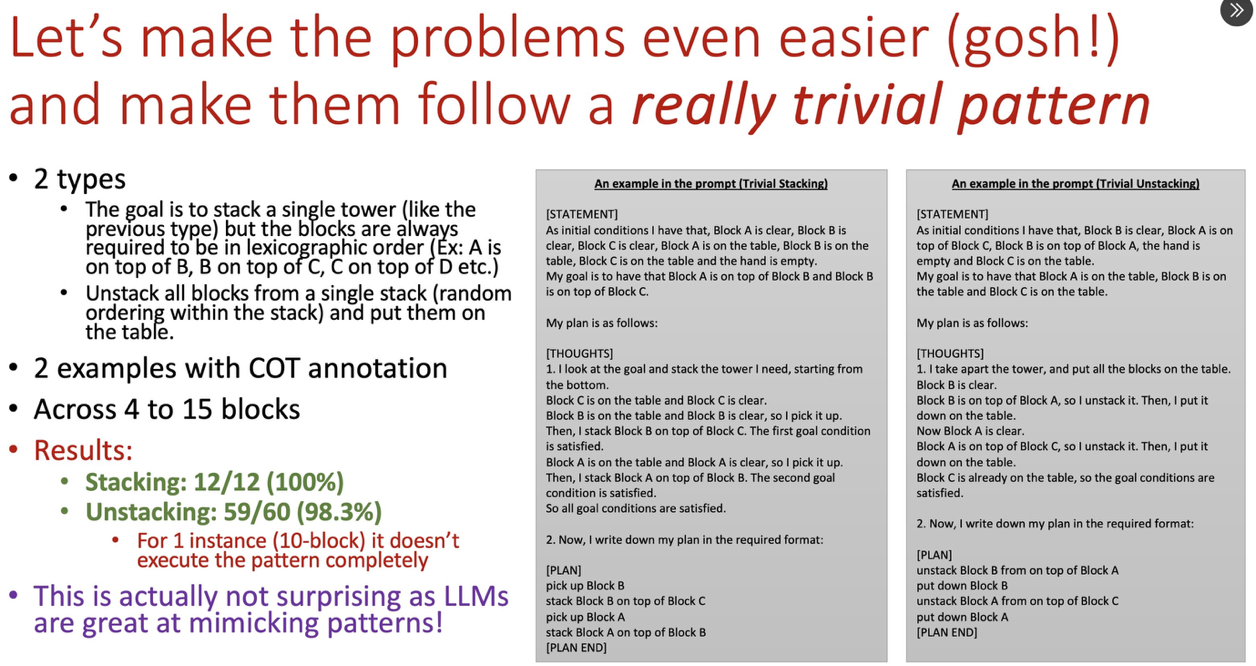
AI-generated content may be incorrect.

At the next level, you might want to give a CoT that is just domain specific. Suppose you are solving a Blocks World problem, it is well known that you can get from any initial state to any goal state by just unstacking and putting all the blocks on the table and using them to stack the various stacks in the goal state. It can be shown that such a plan will be correct and will be at most twice the length of the optimal plan for any instance. Will LLMs be able to run with this CoT? We checked and found that they pretty much die. (The second graphic below👇)

A graph with a line and a line

AI-generated content may be incorrect.

At the third level, you might want to give a CoT that is goal specific. In the case of Blocks World problems, suppose we stick to the same exact problem type: Initial state has all blocks on the table. Goal state has to construct a particular single stack of blocks. You notice that by now, the CoT is quite simple! Even some of you skeptics might think that LLMs will run with this and solve all n-stack problems. We checked and turns out that as the stack length increases, they fail pretty spectacularly. (The third graphic below 👇). The problem is that they don't learn the procedure in stacking and unroll it as needed, and instead do well only if the new instance is close to the example instances in a syntactic (and non-unrolled way).



At the bottom most level, you might just throw in the towel and say that we will focus on a very specific syntactic form of the goal that will avoid LLM having to reason about unrolling. We can do this in blocks world n-stack problems if we make the goal state always be a stack that is a lexicographic prefix. That is, A, AB, ABC, ABCD, ABCDE etc. In this case, finally, CoT actually is kind of effective in helping LLMs solve the n-stack problem. \*PHEW\* (the last graphic below)

As you can see, the ease of giving CoT advice worsens drastically as we go from domain independent to domain specific to goal class specific to lexicographic goal-specific. If you have to teach a student how to do blocks world planning by the bottom level CoT's, you would expel them from school. And yet, CoT has attained a rather mythical status in LLM literature. It shows how willing the practitioners are to suspend their disbelief.

Among other things, the "generalization" claims surrounding CoT cheapen the notion of generalization beyond recognition.