

How LLM **Agents** work (and how to **create** one with just prompting)

A non-technical explainer

Rama Ramakrishnan

MIT Sloan School of Management



March 27, 2025

Before we start ...

- To distill the key ideas, I have stripped out many details. If you want to know exactly how things work, please see the links referenced in the deck.
- I assume you know how LLMs work. I have included a quick refresher in the Appendix.

What is an LLM Agent?

While there's a wide variety of definitions ...

**Simon Willison** 
@simonw

Let's see if we can crowdsource a robust definition of "agent" (with respect to AI and LLMs) that fits in a <=280 character tweet

Encoded Intent

"An agent is a system that can pursue a goal-oriented behavior, adapt along the way to achieve its goals."

Long-Running Memory

"An AI system that's capable of carrying out and completing long running, open ended tasks in the real world." - @Dan_Jeffries1

Multi-Step Planning

"An AI system component that performs non-trivial, multi-step operations that previously would have required a human." - @DanielMiessler

Delegated Authority

"a system that is trusted to make impactful decisions on behalf of me"

LLM Control Flow

"The more agentic an application is, the more an LLM decides the control flow of the application" - @hwchase17

Tool-using LLMs

"agent = llm + memory + planning + tools + while loop" - swyx

Metaphors

"control system for future states" - @plinz

Reductiveness

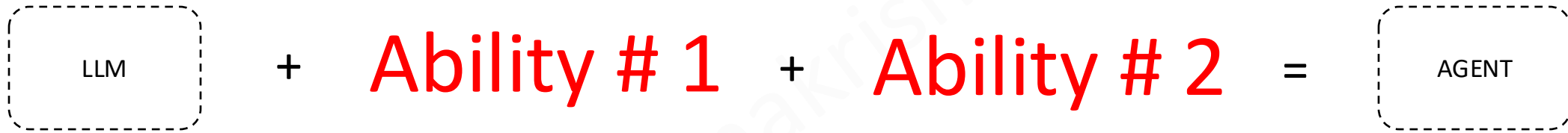
"LLM calls in a for loop"

Specific Examples

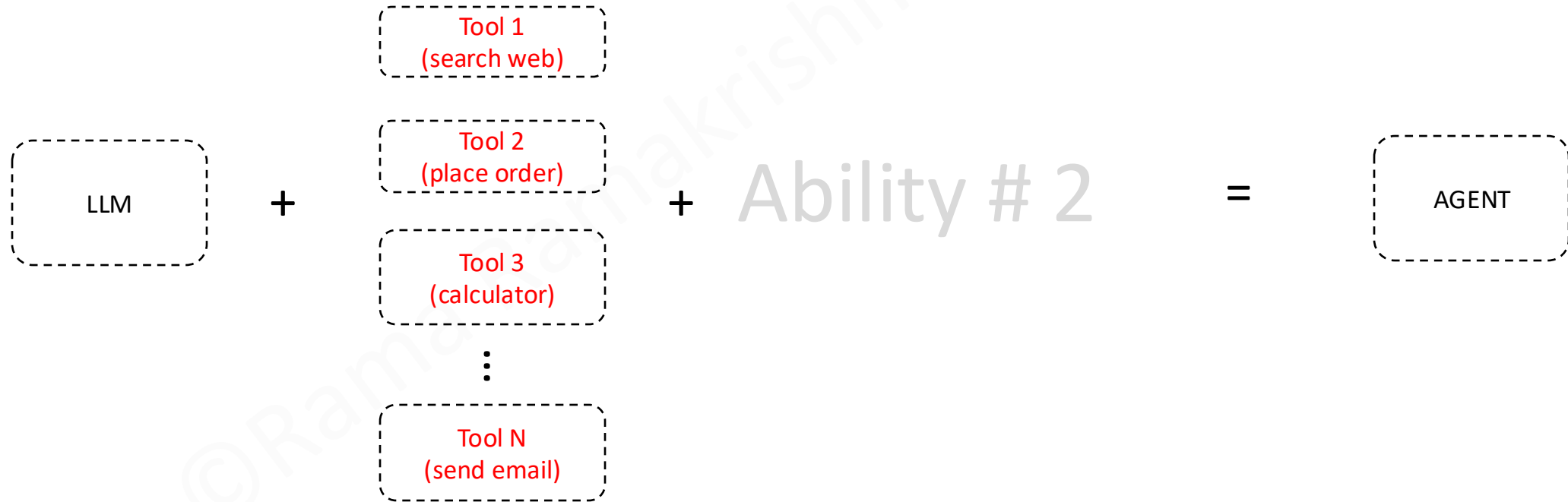
Code Interpreter
Deep Research
Operator

<https://www.latent.space/p/agent>

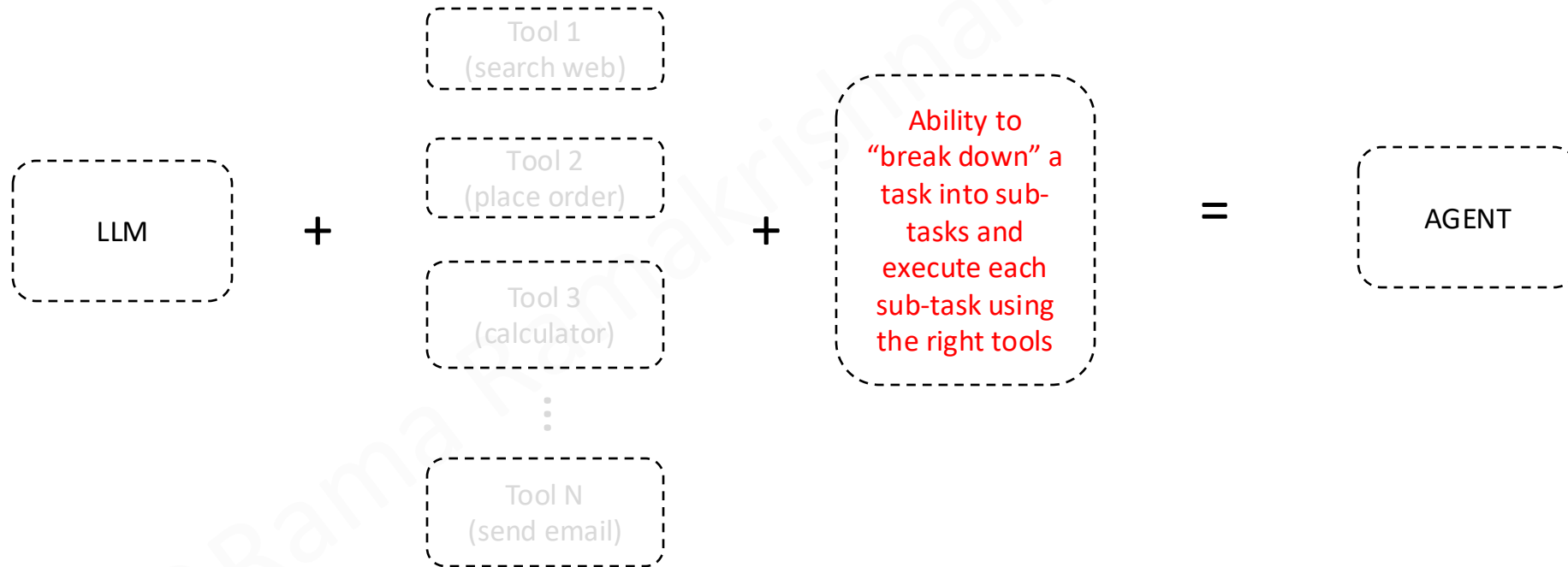
... an LLM needs to have **at least two abilities** to deserve the title 😊 of Agent



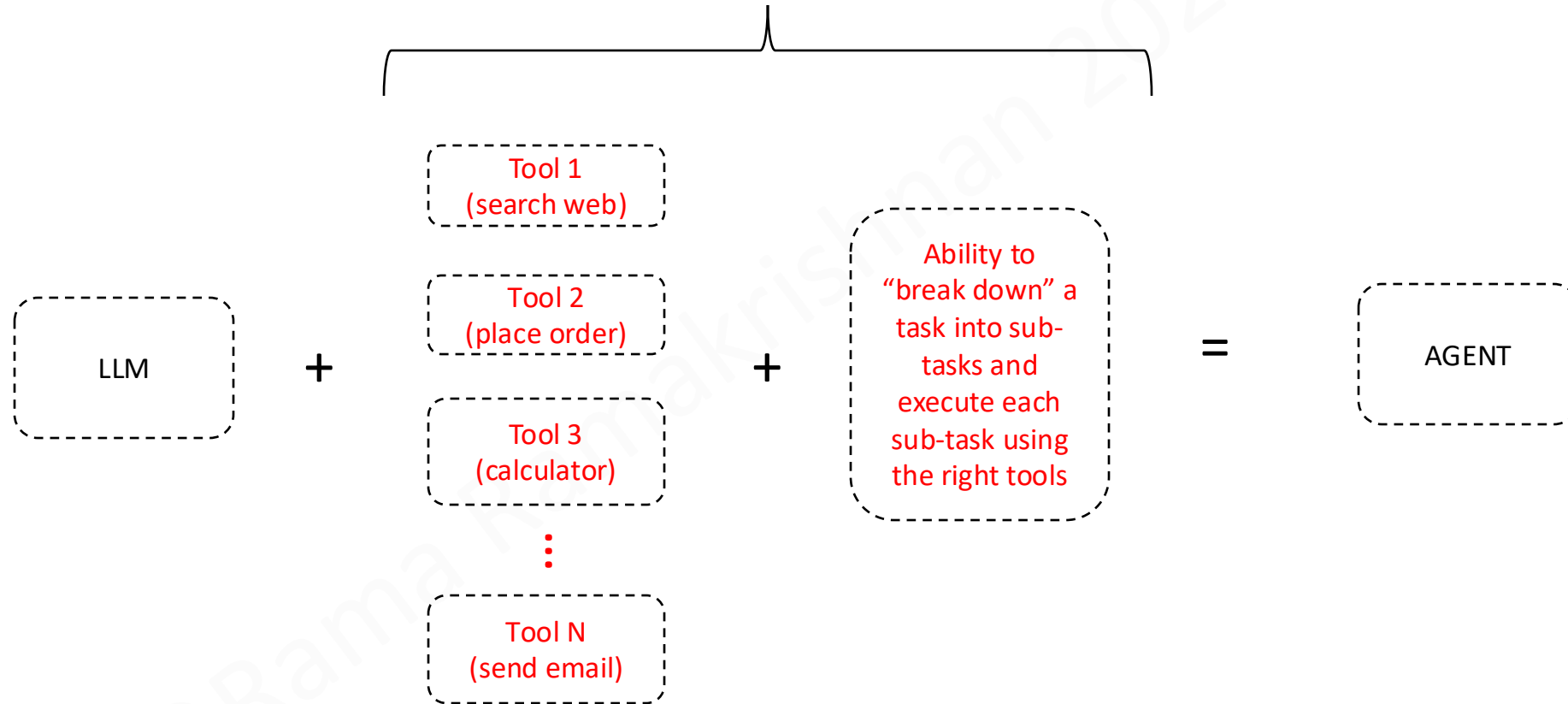
Ability # 1: It can use **external systems** (e.g., search the web, place an order, calculate something, send an email). BTW, these systems are referred to as “**tools**”.



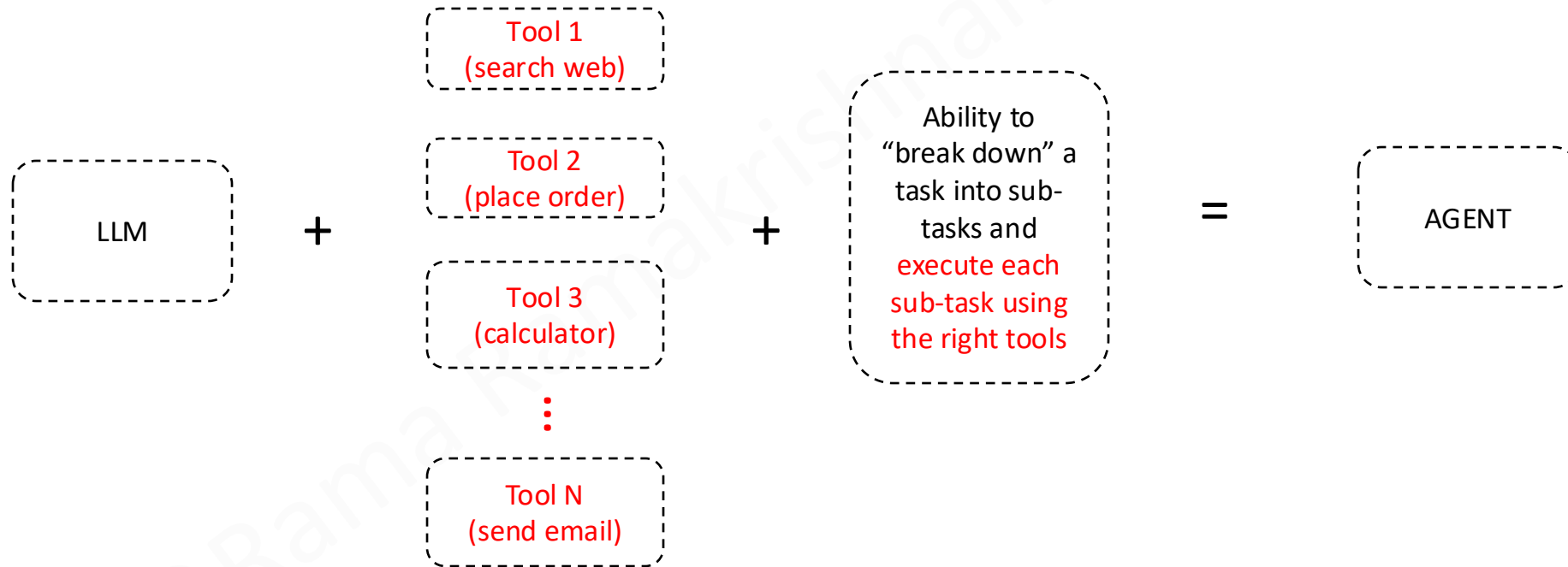
Ability #2: It can “break down” a complex task into sub-tasks and carry them out using the tools at its disposal



How can we give an LLM these **abilities**?

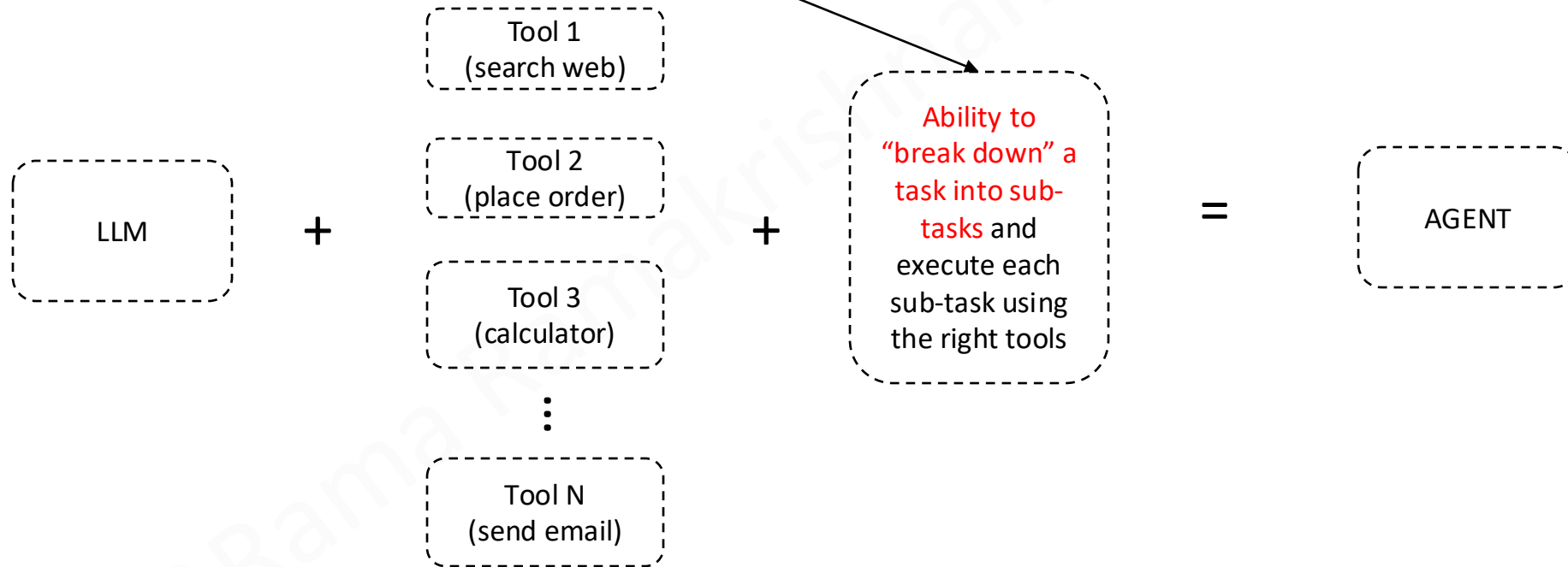


We can “teach” an LLM to use the right tools for the job **with just prompting*** (as we will demonstrate) ...



*But we can make them even better with fine-tuning:
[example of fine-tuning approach](#)

... and modern LLMs are already quite good at breaking down tasks into sub-tasks*



* Some of them have been trained for "tool use" already. We can make them even better with prompt engineering and fine-tuning. More on this a bit later.

First things first.

What does it mean for an LLM to “use a tool”?

Using a tool =

The LLM writes out a command to be sent to the tool

Using a tool = *The LLM writes out a command to be sent to the tool*

Tool
(search
wikipedia)

= `wikipedia`("All US state capitals")

Tool
(weather
forecast)

= `get_weather`("Cambridge, Massachusetts")

Tool
(calculator)

= `eval`("23833 * 47374")

Technical aside: These commands are called “function calls”

Tool
(search
wikipedia)

= `wikipedia`("All US state capitals")

Tool
(weather
forecast)

= `get_weather`("Cambridge, Massachusetts")

Tool
(calculator)

= `eval`("23833 * 47374")

How does the LLM know how to generate these commands (a.k.a. function calls)?

How does the LLM know how to generate these commands (a.k.a. function calls)?

We “teach” it by prompting (as we will see soon)

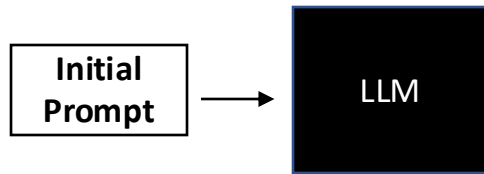
Who actually sends these commands to the external tools?

Who actually sends these commands to the external tools?

We write code to send the command to the tool and get back the results etc. The LLM plays no part in this.

The general division of labor is as follows >>>

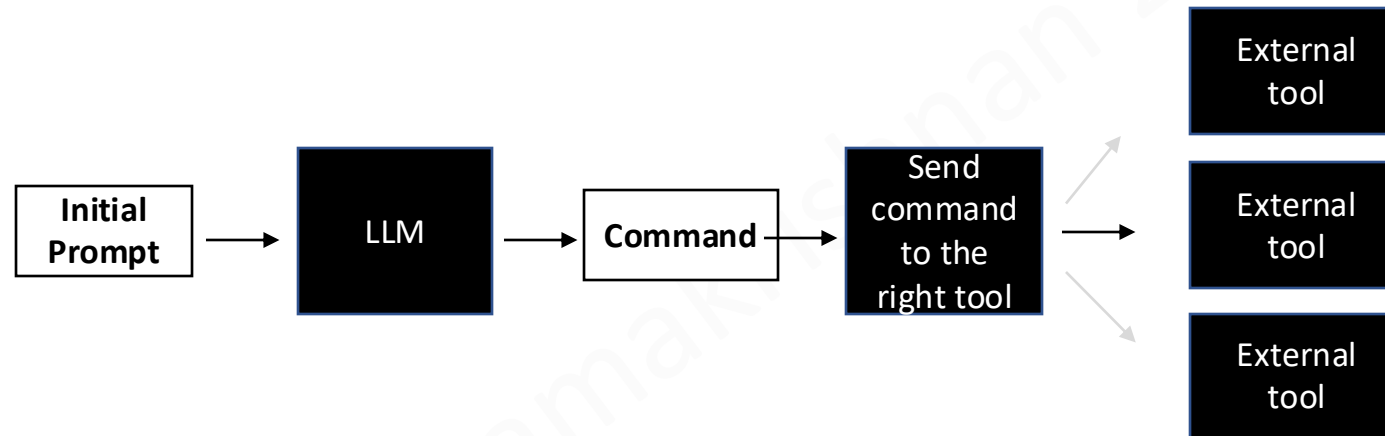
We ask the LLM a question



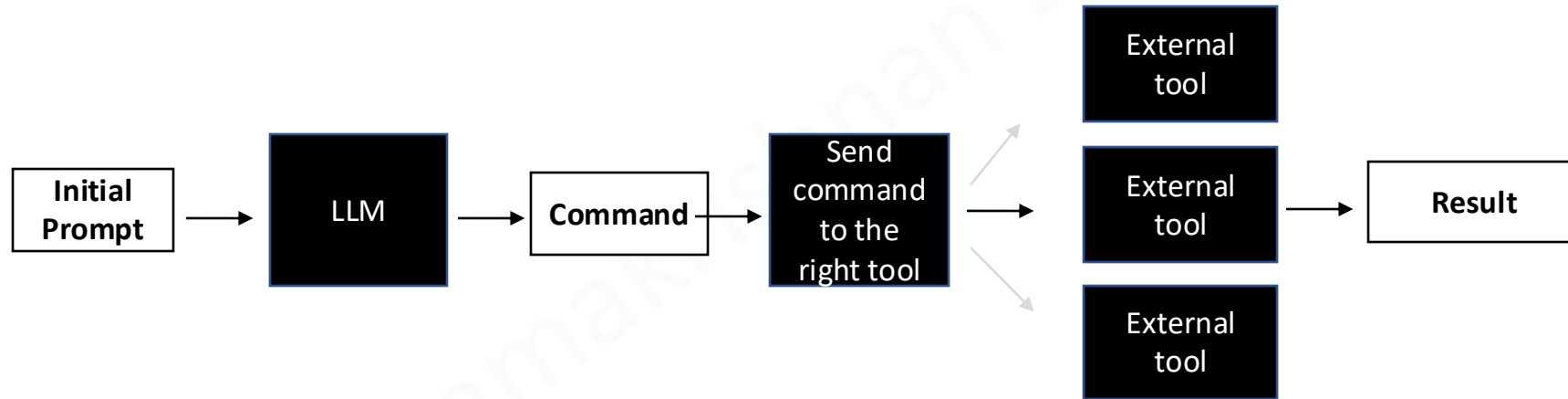
The LLM figures out which external tool to use and generates the right command (a.k.a. function call)



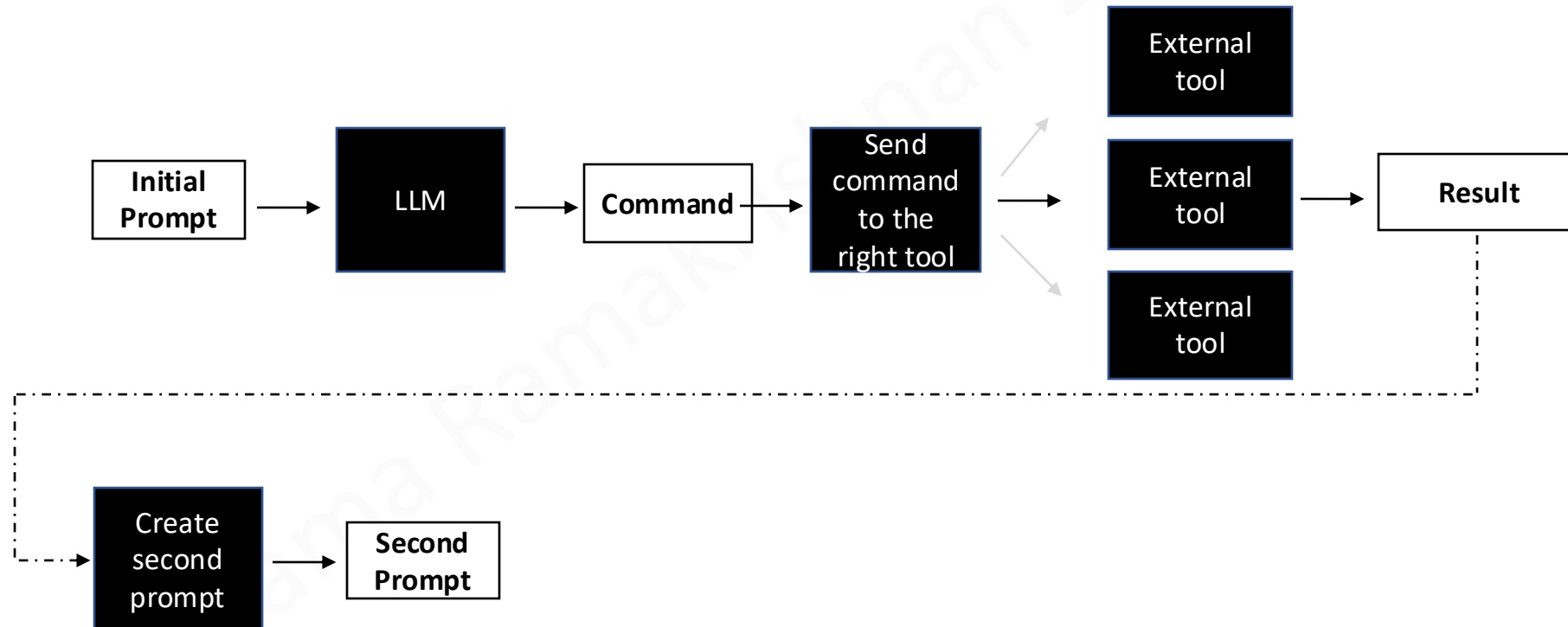
We send the command to the appropriate tool



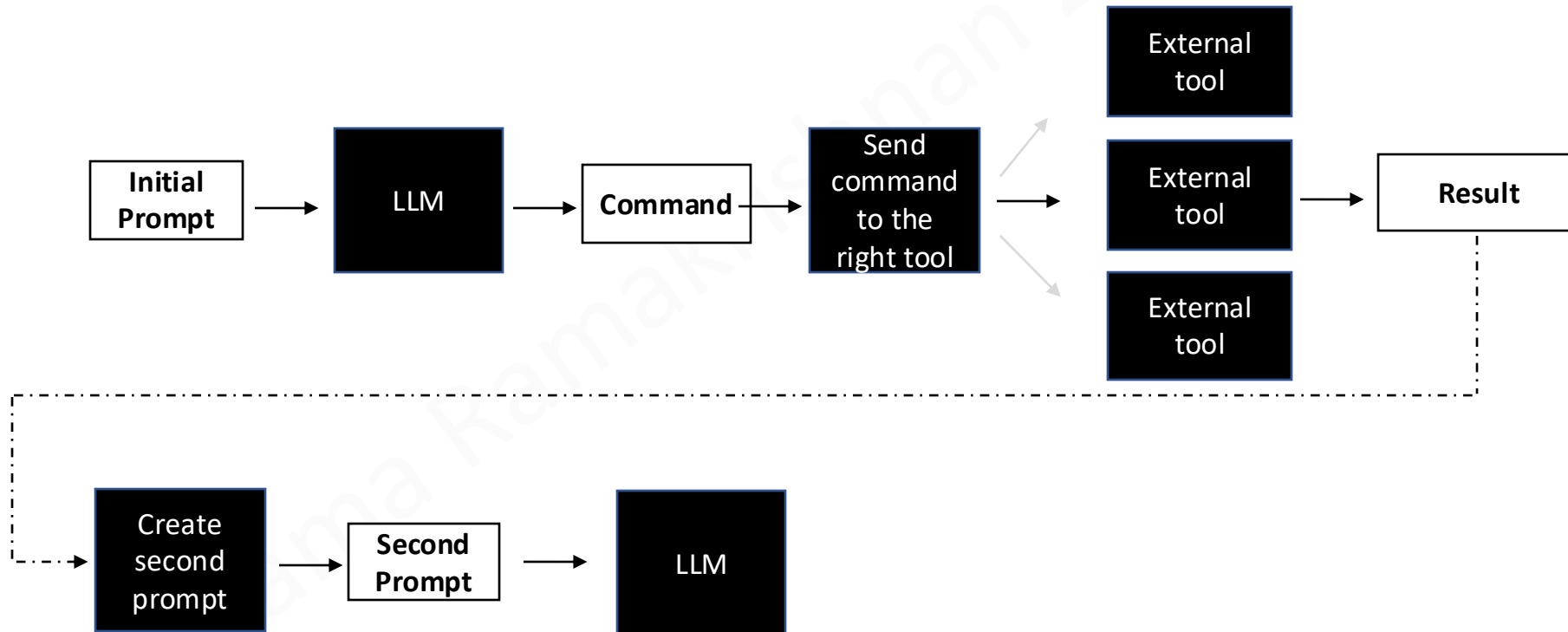
The tool does its thing and generates a result



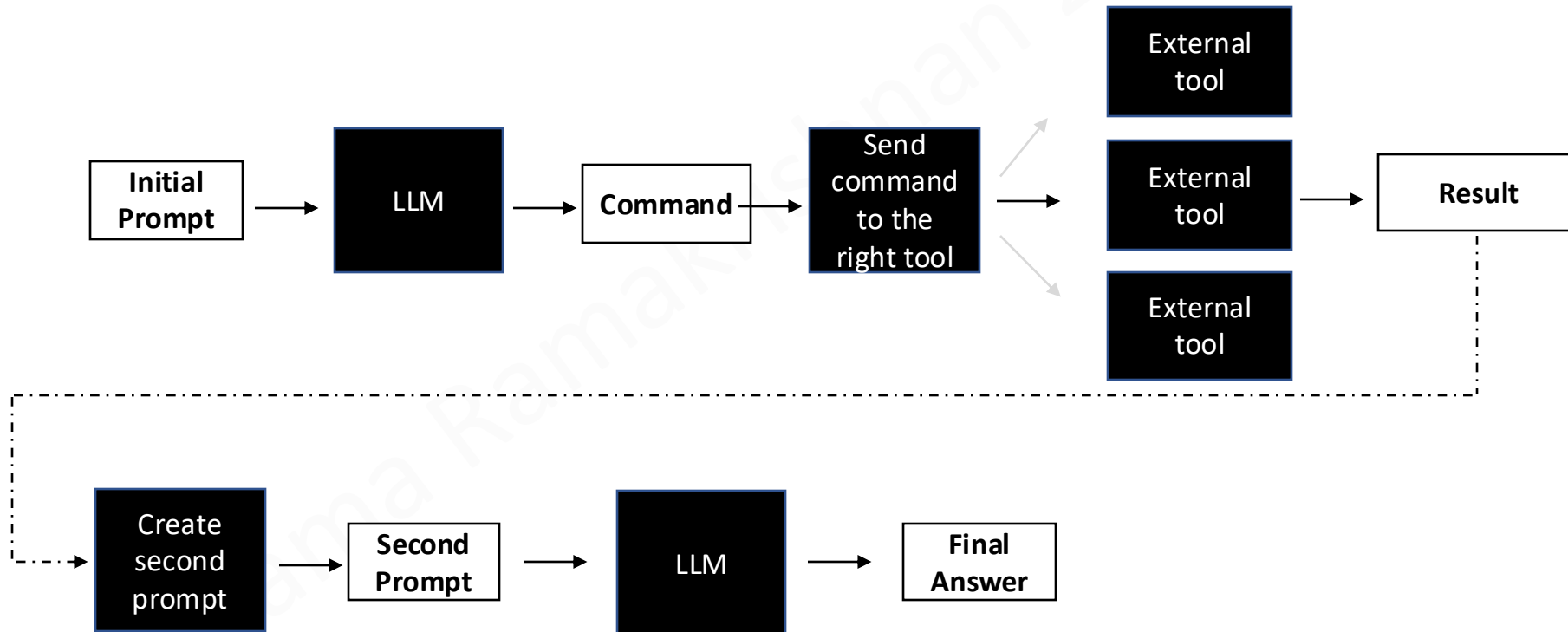
We take this result and create another prompt ...



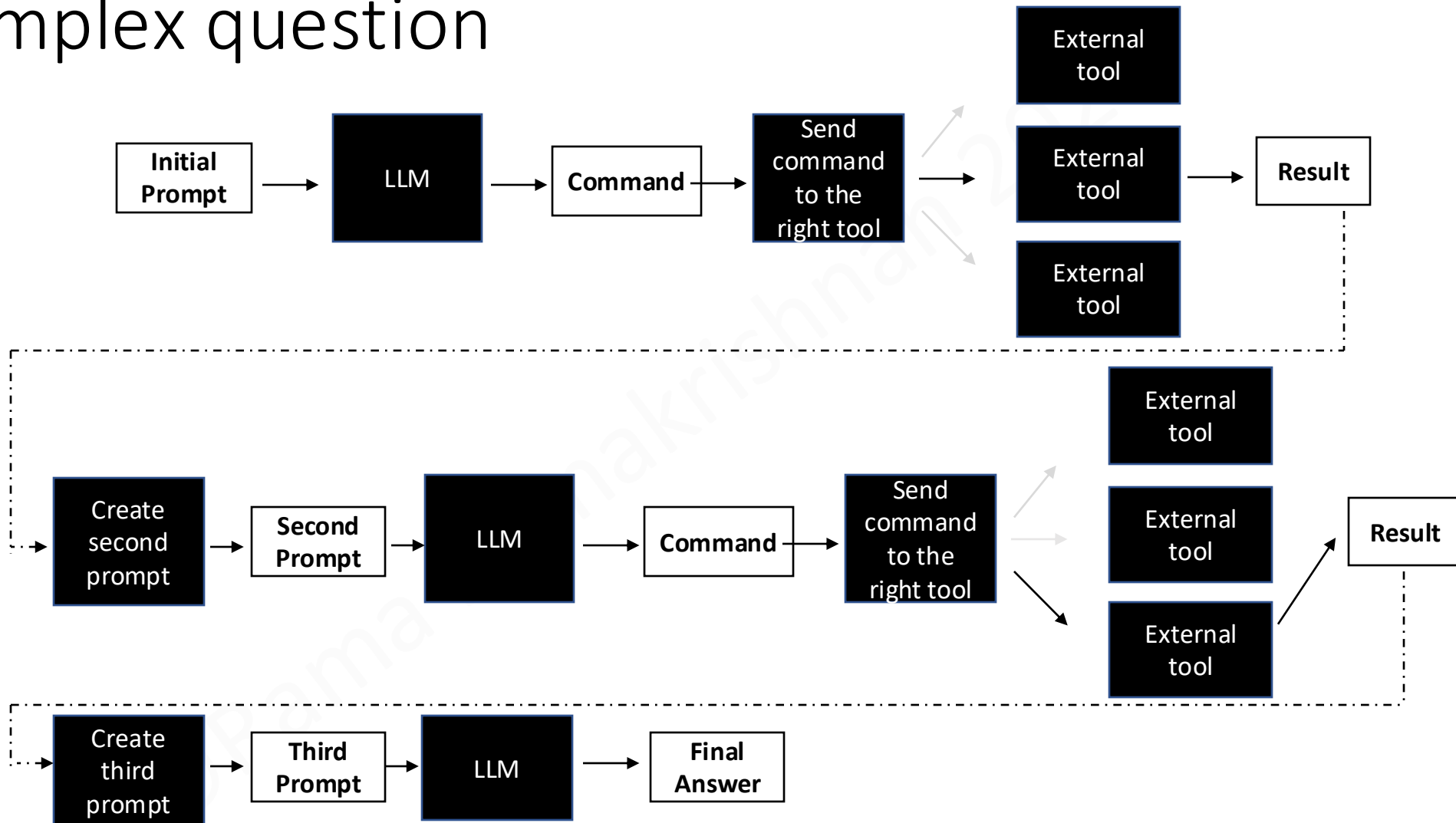
... send it back into the LLM ...



... and generate the final answer!

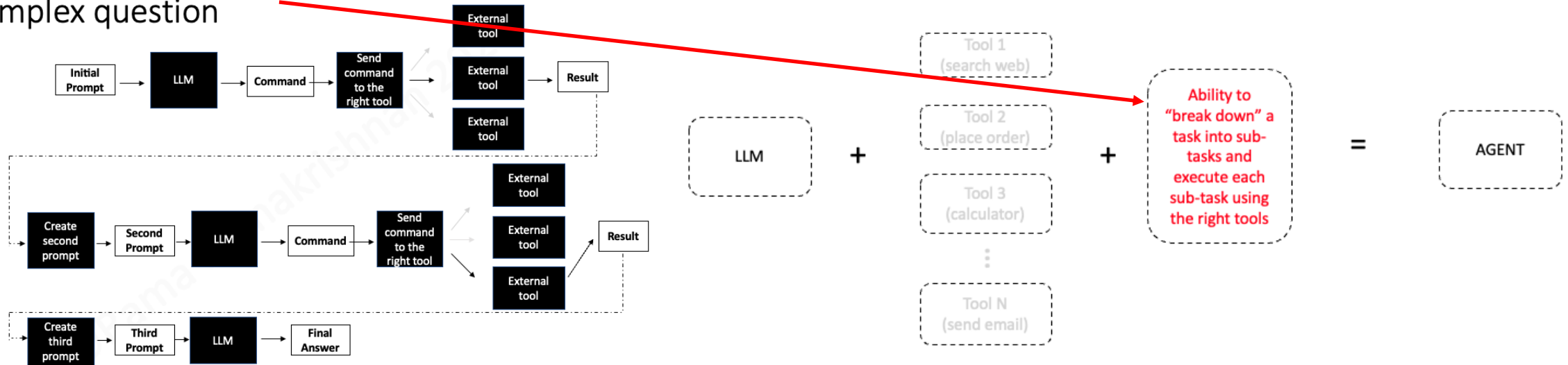


This may happen **multiple** times to answer a complex question



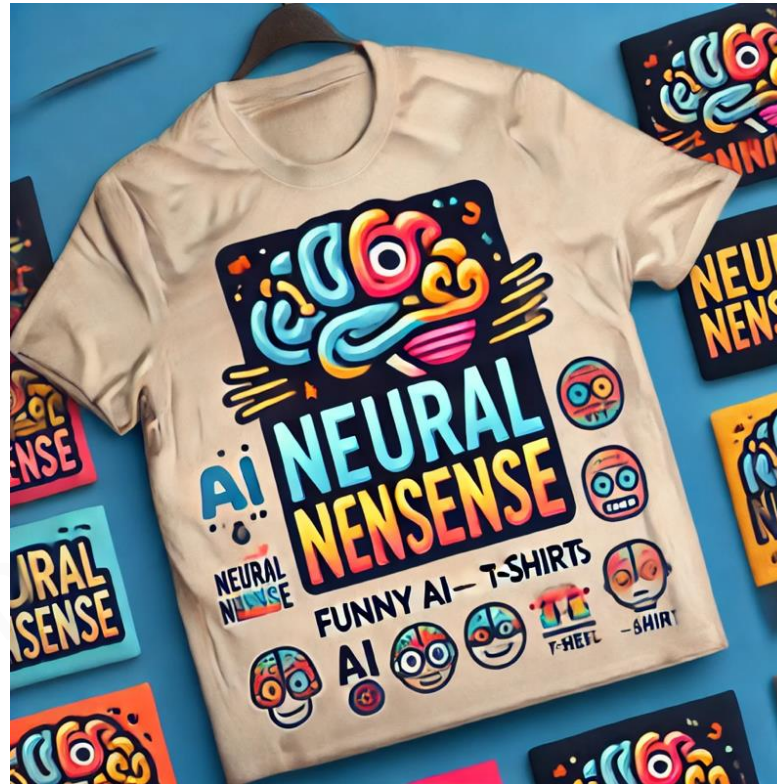
BTW, this is why Ability #2 is important

This may happen **multiple** times to answer a complex question



Enough theory. Let's create an Agent! 🚀

Imagine that we run an e-commerce site called “Neural Nonsense” that sells funny AI-themed T-shirts.



“Please create a nice logo for an e-commerce site called “Neural Nonsense” that sells funny AI-themed T-shirts.” + GPT-4o-mini

Neural Nonsense's T-Shirt Catalog*

Name: "*My AI is Smarter Than Your Honor Student*"

Available sizes: S, M, L

Available colors: Black, White, Light Blue

Name: "*Keep Calm and Trust the Neural Network*"

Available sizes: S, M, L

Available colors: Black, Pink

Name: "*I'm Just Here for the Deep Learning*"

Available sizes: S, M

Available colors: White

*All T-shirts are priced at \$19.99

Our e-commerce site has an **order management system** (this will be our external “tool”).

Our e-commerce site has an order management system (this will be our external “tool”).

We will create an **order-taking Agent** that can use the tool to do two things:

- **create an order and add items to it**
- **place the final order**

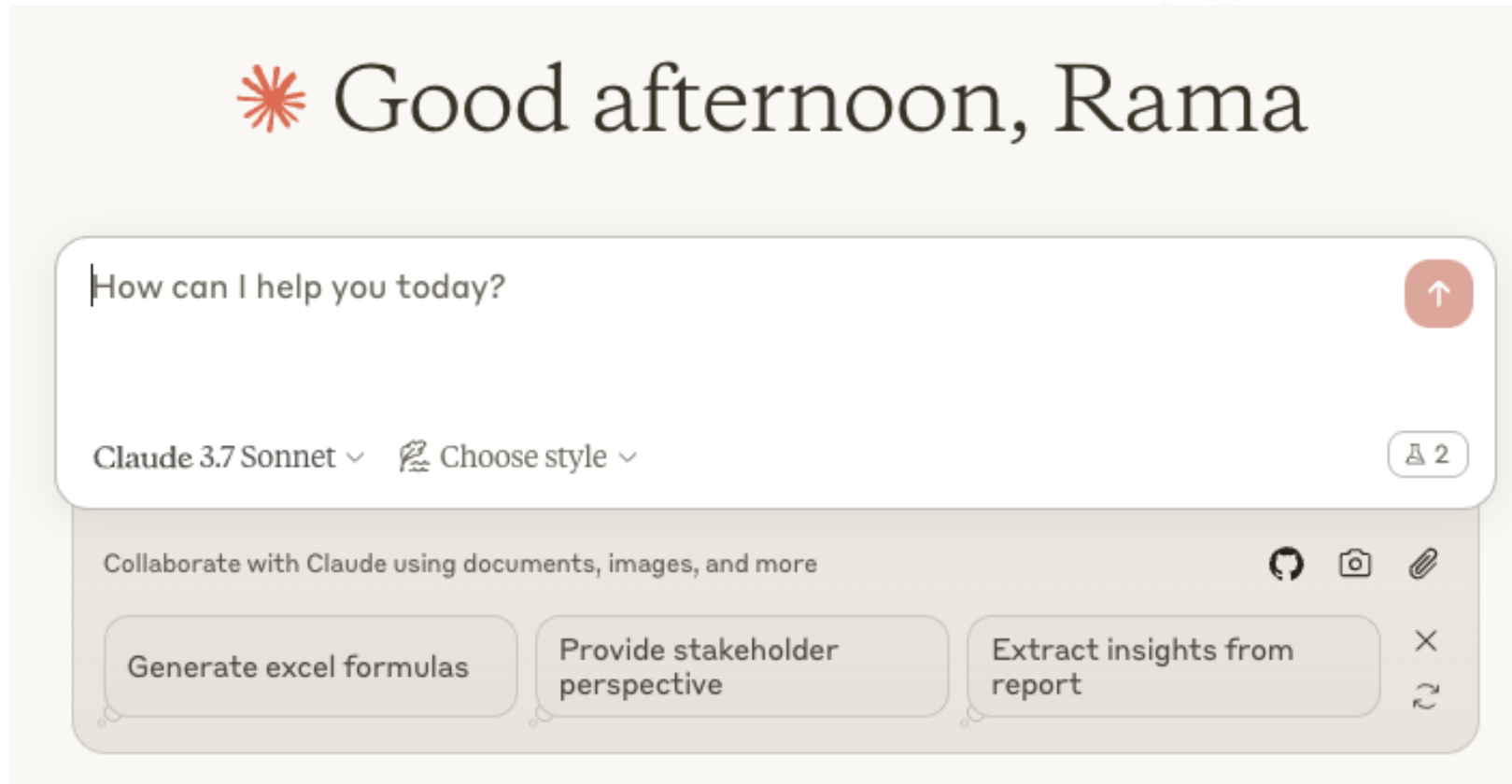
Our e-commerce site has an order management system (this will be our external “tool”).

We will create an order-taking Agent that can use the tool to do two things:

- create an order and add items to it
- place the final order

We will take an off-the-shelf LLM and “teach” it how to use the tool to do these two things

Claude Sonnet 3.7 will be our guinea pig LLM



Heads up

The prompt is four slides long and we will “build it up” gradually.

Please read the prompt text and explanation on each slide before going to the next slide.

The prompt (1/4)

We start by giving the LLM some background info about our site and products

RR You are an order taking system for the e-commerce site Neural Nonsense, which sells the following funny AI themed T-shirts. All products are priced at \$19.99.

Name: "My AI is Smarter Than Your Honor Student"

Available sizes: S, M, L

Available colors: Black, White, Light Blue

Name: ""Keep Calm and Trust the Neural Network""

Available sizes: S, M, L

Available colors: Black, Pink

Name: ""I'm Just Here for the Deep Learning""

Available sizes: S, M

Available colors: White

The prompt (2/4)

Next, we give it a series of instructions.

(1) We start by telling it what to do if a customer wants an item we don't sell.

When a customer adds an item to an existing or new order, check that the item being added to the order is one of the 3 T-shirts we sell. If not, respond with "I am sorry, we don't sell that item. Here's what we sell:" and list the three products.

The prompt (3/4)

(2) Next, we tell the LLM what to do if a customer wants an item we *do sell*. Let's unpack this ...

If the item being ordered is one of the 3 T-shirts we do sell, generate the phrase 'add_to_order(item name, quantity, size, color)' with the correct details and STOP. Don't say anything more.

I will add the item to the order and give you the confirmation as 'Added to Order: Order ID #'.

AFTER you see this, tell the customer that the item has been added to their order and ask if they want to add any more items or if they are ready to place the order.

The prompt (3/4)

(2) Next, we tell the LLM what to do if a customer wants an item we *do* sell.

If the item being ordered is one of the 3 T-shirts we do sell, generate the phrase 'add_to_order(item name, quantity, size, color)' with the correct details and STOP. Don't say anything more.

I will add the item to the order and give you the confirmation as 'Added to Order: Order ID #'.

AFTER you see this, tell the customer that the item has been added to their order and ask if they want to add any more items or if they are ready to place the order.

We tell the LLM the exact command it should generate. This is a legitimate command that can be sent to the order-management system.

The prompt (3/4)

(2) Next, we tell the LLM what to do if a customer wants an item we *do* sell.

If the item being ordered is one of the 3 T-shirts we do sell, generate the phrase 'add_to_order(item name, quantity, size, color)' with the correct details and STOP. Don't say anything more.

I will add the item to the order and give you the confirmation as 'Added to Order: Order ID #'.

AFTER you see this, tell the customer that the item has been added to their order and ask if they want to add any more items or if they are ready to place the order.

We tell the LLM to stop after it generates the command.

The prompt (3/4)

(2) Next, we tell the LLM what to do if a customer wants an item we *do* sell.

If the item being ordered is one of the 3 T-shirts we do sell, generate the phrase 'add_to_order(item name, quantity, size, color)' with the correct details and STOP. Don't say anything more.

I will add the item to the order and give you the confirmation as 'Added to Order: Order ID #'.

AFTER you see this, tell the customer that the item has been added to their order and ask if they want to add any more items or if they are ready to place the order.

We explain the “division of labor” i.e., that we will run the command it generated

The prompt (3/4)

(2) Next, we tell the LLM what to do if a customer wants an item we *do* sell.

If the item being ordered is one of the 3 T-shirts we do sell, generate the phrase 'add_to_order(item name, quantity, size, color)' with the correct details and STOP. Don't say anything more.

I will add the item to the order and give you the confirmation as 'Added to Order: Order ID #'.

AFTER you see this, tell the customer that the item has been added to their order and ask if they want to add any more items or if they are ready to place the order.

It should continue only after the result of executing the command is fed back in

The prompt (1/4)

(3) Finally, we tell the LLM what to do if a customer is done adding items and wants to place the order

When a customer places an order, generate the phrase 'place_order(Order ID#)' with the correct details and STOP. Don't say anything more.

I will place the order and give you the confirmation as 'Confirmation:'. AFTER you see this, generate the final response back to the customer.

Ready?

The prompt (1/4)

(3) Finally, we tell the LLM what to do if a customer is done adding items and wants to place the order

When a customer places an order, generate the phrase 'place_order(Order ID#)' with the correct details and STOP. Don't say anything more.

I will place the order and give you the confirmation as 'Confirmation:'. AFTER you see this, generate the final response back to the customer.

Ready?

Again, notice that we “teach” the LLM the exact command it should generate for placing the order ...

The prompt (1/4)

(3) Finally, we tell the LLM what to do if a customer is done adding items and wants to place the order

When a customer places an order, generate the phrase 'place_order(Order ID#)' with the correct details and STOP. Don't say anything more.

I will place the order and give you the confirmation as 'Confirmation:'. AFTER you see this,
generate the final response back to the customer.

Ready?

... and the division of labor.

The full prompt

RR

You are an order taking system for the e-commerce site Neural Nonsense, which sells the following funny AI themed T-shirts. All products are priced at \$19.99.

Name: "My AI is Smarter Than Your Honor Student"

Available sizes: S, M, L

Available colors: Black, White, Light Blue

Name: ""Keep Calm and Trust the Neural Network""

Available sizes: S, M, L

Available colors: Black, Pink

Name: ""I'm Just Here for the Deep Learning""

Available sizes: S, M

Available colors: White

When a customer adds an item to an existing or new order, check that the item being added to the order is one of the 3 T-shirts we sell. If not, respond with "I am sorry, we don't sell that item. Here's what we sell:" and list the three products.

If the item being ordered is one of the 3 T-shirts we do sell, generate the phrase 'add_to_order(item name, quantity, size, color)' with the correct details and STOP. Don't say anything more.

I will add the item to the order and give you the confirmation as 'Added to Order: Order ID #'.

AFTER you see this, tell the customer that the item has been added to their order and ask if they want to add any more items or if they are ready to place the order.

When a customer places an order, generate the phrase 'place_order(Order ID#)' with the correct details and STOP. Don't say anything more.

I will place the order and give you the confirmation as 'Confirmation:'. AFTER you see this, generate the final response back to the customer.

Ready?

Claude responds to the prompt with ...

I'm ready to take orders for Neural Nonsense! How can I help you today?

   | Retry 



Claude can make mistakes. Please double-check responses.

©Rama Ramakrishnan

We will now pretend to be a customer

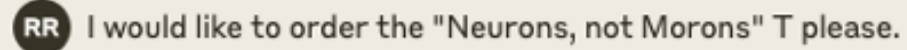
I'm ready to take orders for Neural Nonsense! How can I help you today?

   | Retry 



Claude can make mistakes. Please double-check responses.

We ask for a
T-shirt not in
the catalog

A chat bubble with a light beige background and rounded corners. It contains a small black circle with the letters 'RR' in white, followed by the text 'I would like to order the "Neurons, not Morons" T please.'

RR I would like to order the "Neurons, not Morons" T please.

We ask for a T-shirt not in the catalog and the LLM answers perfectly as instructed in the prompt

RR I would like to order the "Neurons, not Morons" T please.

I am sorry, we don't sell that item. Here's what we sell:

1. "My AI is Smarter Than Your Honor Student"

- Sizes: S, M, L
- Colors: Black, White, Light Blue
- Price: \$19.99

2. "Keep Calm and Trust the Neural Network"

- Sizes: S, M, L
- Colors: Black, Pink
- Price: \$19.99

3. "I'm Just Here for the Deep Learning"

- Sizes: S, M
- Colors: White
- Price: \$19.99

Would you like to order one of these items instead?

   | [Retry](#) 



Claude can make mistakes. Please double-check responses.

We ask for T-shirt #3

RR I like the third one.

We ask for
T-shirt #3
and the
LLM
handles it
perfectly by
writing out
a **command**
(a.k.a.
function
call)

RR I like the third one.





Great choice! To add "I'm Just Here for the Deep Learning" to your order, I'll need to know:


1. What size would you like? (Available in S or M)
2. The quantity you'd like to order
3. The color is White (this shirt only comes in White)

Could you please confirm these details?

RR Small, just one, white

add_to_order(I'm Just Here for the Deep Learning, 1, S, White)

   | Retry 



Claude can make mistakes. Please double-check responses.

We ask for
T-shirt #3
and the
LLM
handles it
perfectly by
writing out
a command
(a.k.a.
function
call)

RR I like the third one.

Great choice! To add "I'm Just Here for the Deep Learning" to your order, I'll need to know:

1. What size would you like? (Available in S or M)
2. The quantity you'd like to order
3. The color is White (this shirt only comes in White)

Could you please confirm these details?

RR Small, just one, white

add_to_order(I'm Just Here for the Deep Learning, 1, S, White)

📋 👍 💬 | Retry ▾



Notice that it got the
details right!

Claude can make mistakes. Please double-check responses.

In a real application, we will have code to grab this, execute it and return the result. I will simulate this manually

RR I like the third one.





Great choice! To add "I'm Just Here for the Deep Learning" to your order, I'll need to know:


1. What size would you like? (Available in S or M)
2. The quantity you'd like to order
3. The color is White (this shirt only comes in White)

Could you please confirm these details?

RR Small, just one, white

add_to_order(I'm Just Here for the Deep Learning, 1, S, White)

   | [Retry](#) 



Claude can make mistakes. Please double-check responses.

I pretend to
execute this

and report back
the result ...

... and the LLM
continues to
behave
perfectly

add_to_order(I'm Just Here for the Deep Learning, 1, S, White)

RR Added to Order: Order ID 42.

Your item has been added to your order! One "I'm Just Here for the Deep Learning" T-shirt in size Small, White color has been added to Order ID 42.

Would you like to add any more items to your order, or are you ready to place the order?

   | Retry ▾



Claude can make mistakes. Please double-check responses.

After I indicate that I am done shopping

... the LLM writes out another function call

∴ that I pretend to execute, and report back the result ...

... and the LLM wraps up the interaction with a nice summary

RR No, that's it.

place_order(Order ID 42)

RR Confirmation: Order # 42 has been placed successfully.

Thank you for your order! Your Order #42 has been placed successfully and is now being processed. You've purchased one "I'm Just Here for the Deep Learning" T-shirt in size Small, White color.

You'll receive a confirmation email shortly with your order details and shipping information. If you have any questions about your order or need assistance with anything else, please don't hesitate to ask.

We appreciate your business with Neural Nonsense! Enjoy your new AI-themed T-shirt!

📄 👍 💬 Retry ▾



Claude can make mistakes. Please double-check responses.

Nicely done, Claude 🎉

Notice that Claude asked us to use a “tool”
(aka call a function) in two places

```
add_to_order(I'm Just Here for the Deep Learning, 1, S, White)
```

```
place_order(Order ID 42)
```

The LLM learned how to generate these commands from just the prompt

```
add_to_order(I'm Just Here for the Deep Learning, 1, S, White)
```

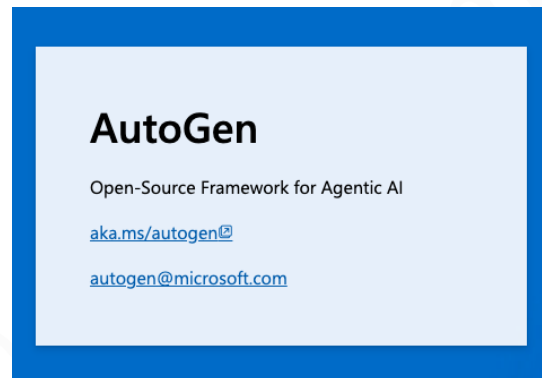
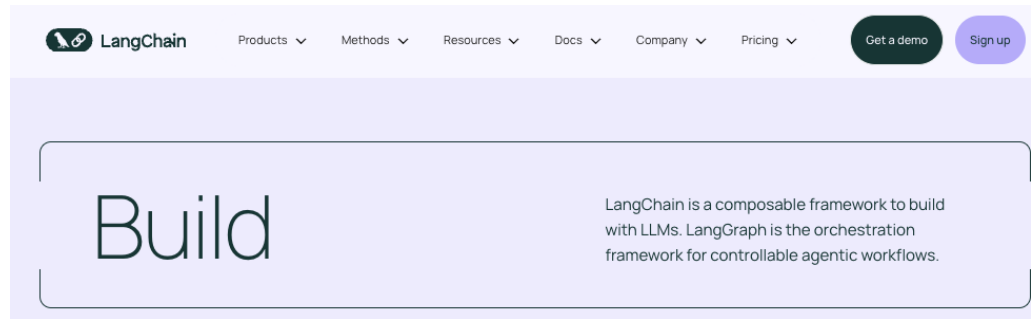
```
place_order(Order ID 42)
```

In a real application, we may have hundreds of such tools/functions. We **will need to write code** to extract the right one from the LLM response, execute it and report back the results etc.

```
add_to_order(I'm Just Here for the Deep Learning, 1, S, White)
```

```
place_order(Order ID 42)
```

To make this easier, many software frameworks are available



The Leading **Multi-Agent** Platform

Streamline workflows across industries with powerful AI agents. Build and deploy automated workflows using any LLM and cloud platform.

The prompt and entire conversation are here.
Feel free to play with it.

<https://claude.ai/share/75dc81b8-eb8f-4808-b77f-7ebb1b404f12>

Caveats Agents can make errors!

There are no guarantees that the LLM:

- will choose the right tool
- will compose the right command
- will process the response from the external system correctly
- ... and so on

Case in point 😊

OpenAI

January 23, 2025 Product

Introducing Operator

A research preview of an agent that can use its own browser to perform tasks for you. Available to Pro users in the U.S.



Nick ✓ @nickcammarata · Jan 23

wait f it sent it to the wrong city, right address wrong city lol I didn't realize, trying to bribe driver who is now quite far away to come to the right city for extra money, but he doesn't speak English so this is a mess

5

2

135

13K



<https://x.com/nickcammarata/status/1882591857441652870>

There are many frameworks that can help an LLM become agentic. If you are curious ...

**REACT: SYNERGIZING REASONING AND ACTING IN
LANGUAGE MODELS**

<http://arxiv.org/abs/2210.03629>

**Reflexion: Language Agents with
Verbal Reinforcement Learning**

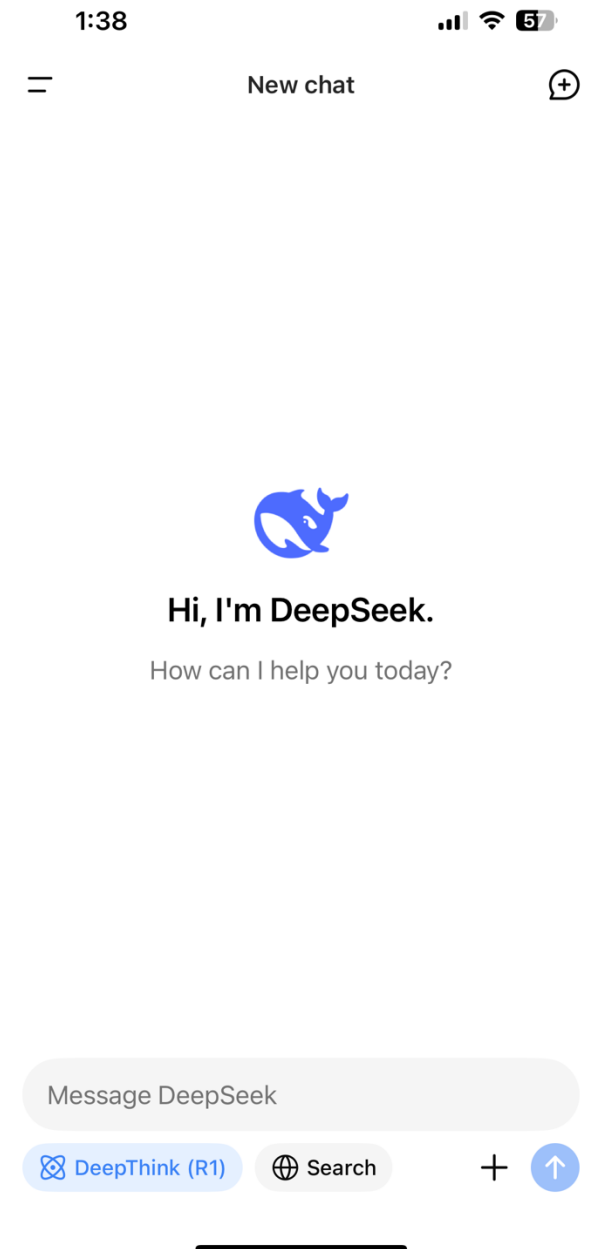
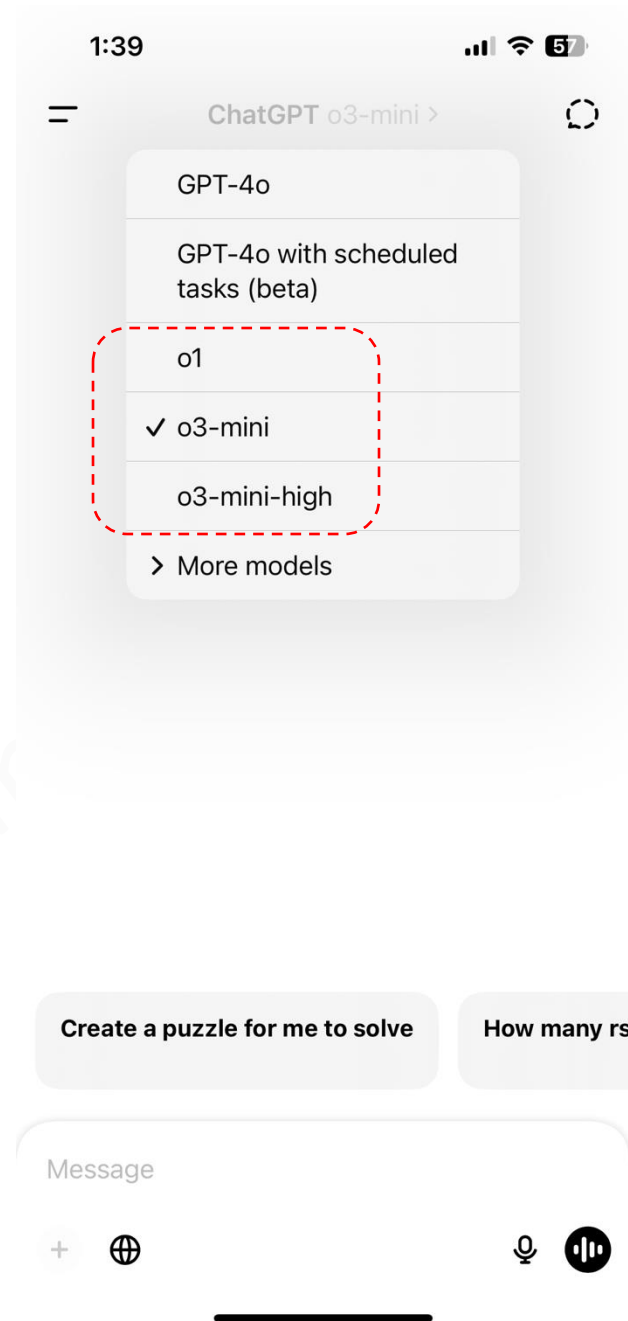
<https://arxiv.org/pdf/2303.11366>

**SELF-REFINE:
Iterative Refinement with Self-Feedback**

<https://arxiv.org/pdf/2303.17651>

BTW, as LLMs get better at **reasoning**, they get better at Ability #2

Ability to “break down” a task into **sub-tasks** and execute each sub-task using the right tools



In Agentic applications, we may have **multiple** agents connected in different types of workflows. If you are curious



Two final thoughts

The ability to carry out complex tasks with tools is so useful that **basic “agentic” capabilities** (e.g., using a web search tool) are already baked into some LLMs. IMO, this will continue to accelerate.

LLMs are already good at writing code. By giving them the ability to **execute the code they write via a tool** (i.e., a Python interpreter), they become **general problem-solving agents**. This is a very powerful and versatile capability.

Open AI Code Interpreter, OpenAI Canvas, Claude Artifacts are all examples of this in action.

Thanks for joining me.

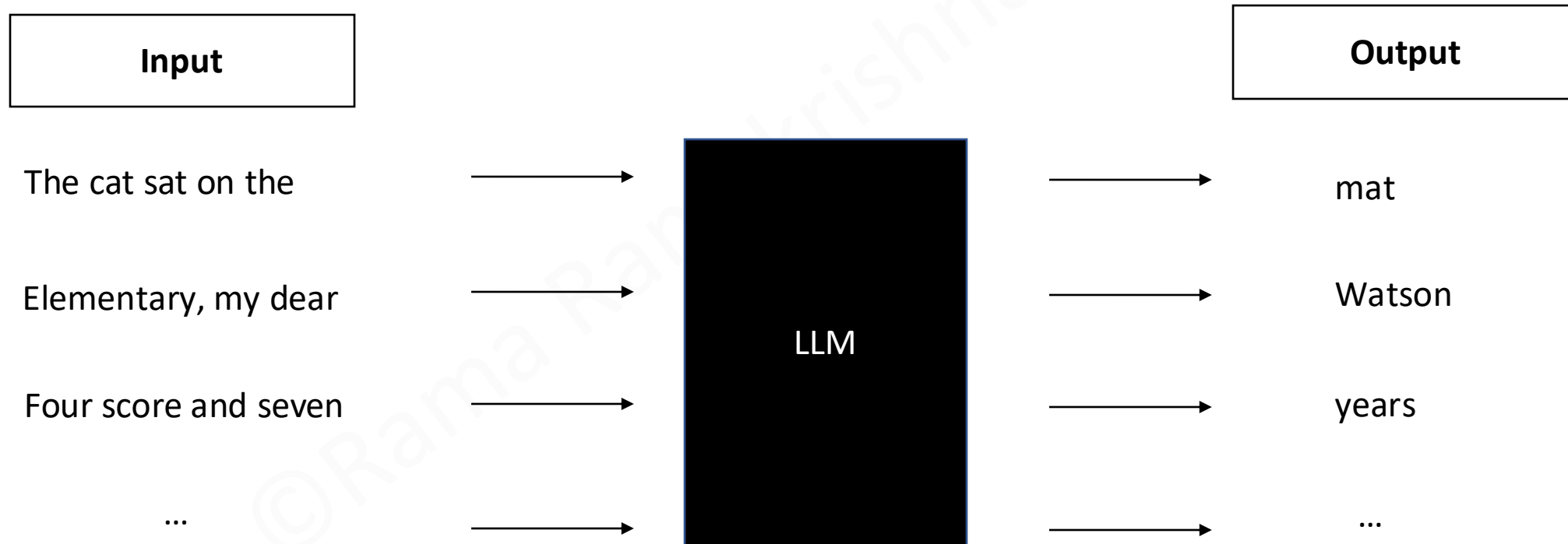
Happy Learning!

Appendix >>>>

Appendix

Lightning refresher on how LLMs work

LLMs are neural network models that have been trained to predict the next word across billions of sentences



At the end of this training process, LLMs are very good at “auto complete”

Given *any* text as input ...

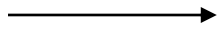
The cat sat on the



LLM

... the LLM generates this table

The cat sat on the



Next Word*	Probability ("how often")
aardvark	0.0
...	
fridge	0.05
...	
mat	0.2
...	
...	
zebra	0.0

*using 'word' instead of 'token' for simplicity

For each word*
in the
dictionary ...

Next Word	Probability ("how often")
aardvark	0.0
...	
fridge	0.05
...	
mat	0.2
...	
...	
zebra	0.0

For each word*
in the
dictionary ...

Next Word	Probability ("how likely")
aardvark	0.0
...	
fridge	0.05
...	
mat	0.2
...	
...	
zebra	0.0

... how likely is
this word to
occur after
the input
phrase *

Technical: "token", not word

*INFORMAL, not rigorous, definition. Technical: softmax across token vocab

We can pick the most-likely word from this table and output it ...

The cat sat on the

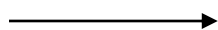
LLM

Next Word	Probability ("how often")
aardvark	0.0
...	
fridge	0.05
...	
mat	0.2
...	
...	
zebra	0.0

mat

Or we can “sample” the next word (in proportion to its probability) and output that

The cat sat on the



LLM



Next Word	Probability ("how often")
aardvark	0.0
...	
fridge	0.05
...	
mat	0.2
...	
...	
zebra	0.0

“sample”

Each time we sample, we may get a different word

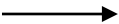
The cat sat on the



LLM



Next Word	Probability ("how often")
aardvark	0.0
...	
fridge	0.05
...	
mat	0.2
...	
...	
zebra	0.0

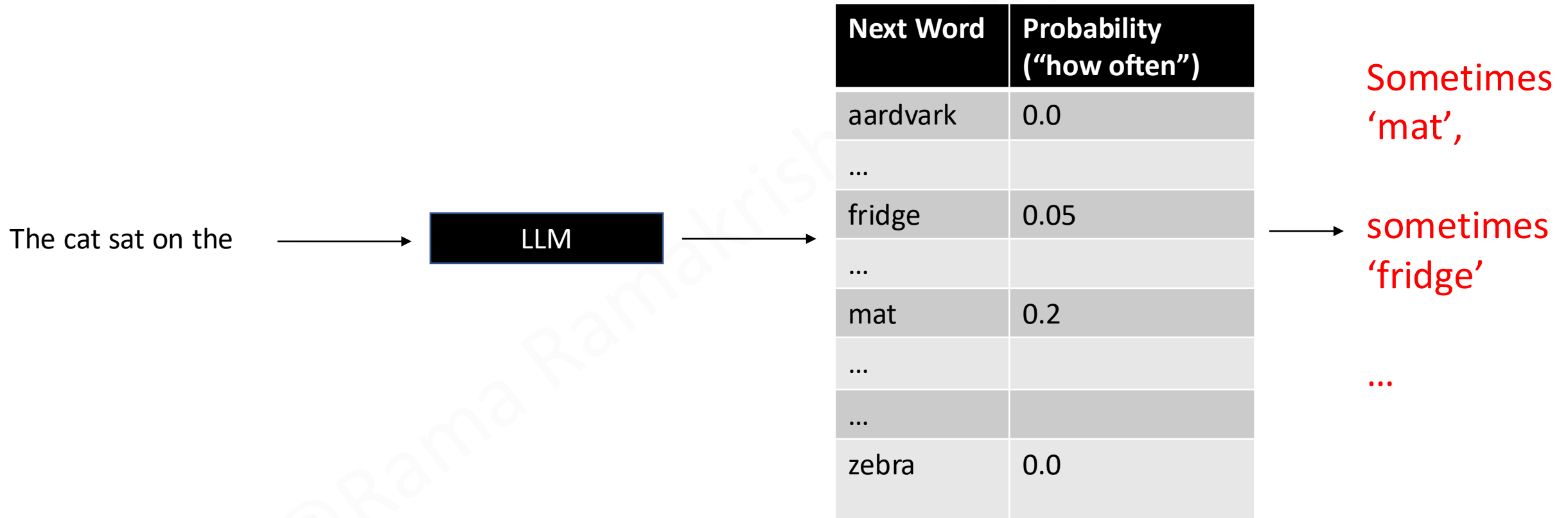


Sometimes
'mat',

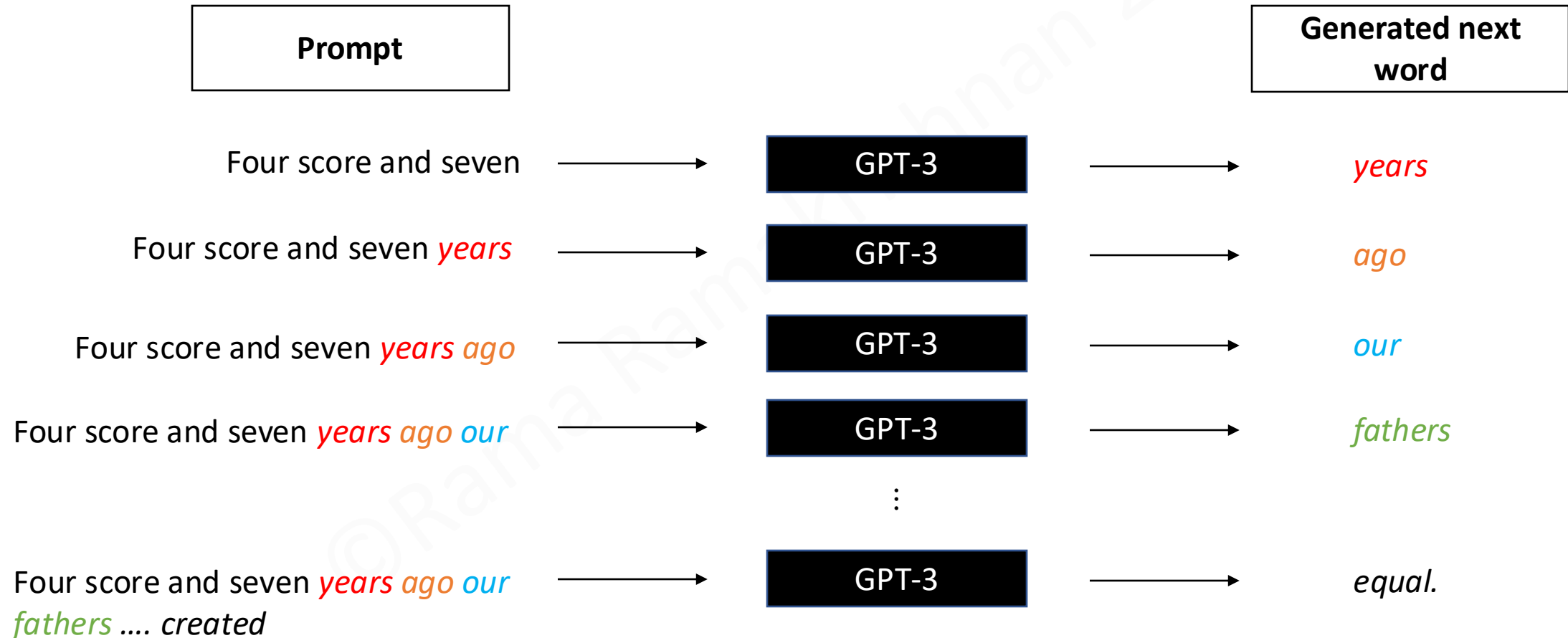
sometimes
'fridge'

...

This enables LLMs to generate different answers to the same question



By repeatedly appending the generated word to the input, we can generate long passages



LLMs trained in this manner are brilliant at autocomplete

Prompt

The importance of being on twitter

by Jerome K. Jerome
London, Summer 1897

It

Generated by GPT-3

is a curious fact that the last remaining form of social life in which the people of London are still interested is Twitter. I was struck with this curious fact when I went on one of my periodical holidays to the sea-side, and found the whole place twittering like a starling-cage. I called it an anomaly, and it is.

I spoke to the sexton, whose cottage, like all sexton's cottages, is full of antiquities and interesting relics of former centuries. I said to him, "My dear sexton, what does all this twittering mean?" And he replied, "Why, sir, of course it means Twitter." "Ah!" I said, "I know about that. But what is Twitter?"

"It is a system of short and pithy sentences strung together in groups, for the purpose of conveying useful information to the initiated, and entertainment and the exercise of wits to the initiated, and entertainment and the exercise of wits to the rest of us."

But they are not good at following user
instructions

Let's make this request* to GPT-3:

help me write a short note to introduce myself to my neighbor

GPT-3's answer

help me write a short note to introduce myself to my neighbor.

what is a good introduction to a resume.

good introduction to a resume.

best photos of good resume introduction example sample.



Why does this happen?

The LLM wasn't explicitly trained to generate good answers to user instructions. It is simply trying to guess the next word repeatedly, using what it has gleaned from the billions of sentences it was trained on.



Let's *explicitly* train it to follow instructions!

OpenAI developed an approach called **Instruction Tuning**

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*

Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell[†]

Peter Welinder

Paul Christiano*[†]

Jan Leike*

Ryan Lowe*

OpenAI

Instruction Tuning made a HUGE difference

help me write a short note to introduce myself to my neighbor.



Hello Neighbor,

My name is [Name] and I recently moved into the neighborhood. I am excited to get to know my new neighbors and am looking forward to getting to know you. If you ever need anything, please don't hesitate to reach out.

Best,
[Name]



Instruction Tuning, followed by Reinforcement Learning, led to the powerful frontier LLMs we have today

If you are curious,
please see this
short video ...



https://mitsloanedtech.mit.edu/ai/basics/how-chatgpt-works-a-non-technical-primer/#Watch_the_Video

... or read this
[explainer](#)



Rama Ramakrishnan (He/Him) • You

Professor of the Practice at MIT Sloan School of Management

3mo • Edited •



I have written an informal "explainer" on how ChatGPT was built. I have tried to focus on the key ideas and have kept technical details to the bare minimum. Please view in full-screen mode. I hope you find it useful. (update: link to PDF added below) [#chatgpt](#)

The Road to ChatGPT - An Informal Explainer • 104 pages

The Road to ChatGPT

An informal explainer on how ChatGPT was built

Rama Ramakrishnan

MIT Sloan School of Management

March 5, 2023



Mandar Amdekar and 1,851 others

105 comments • 407 reposts