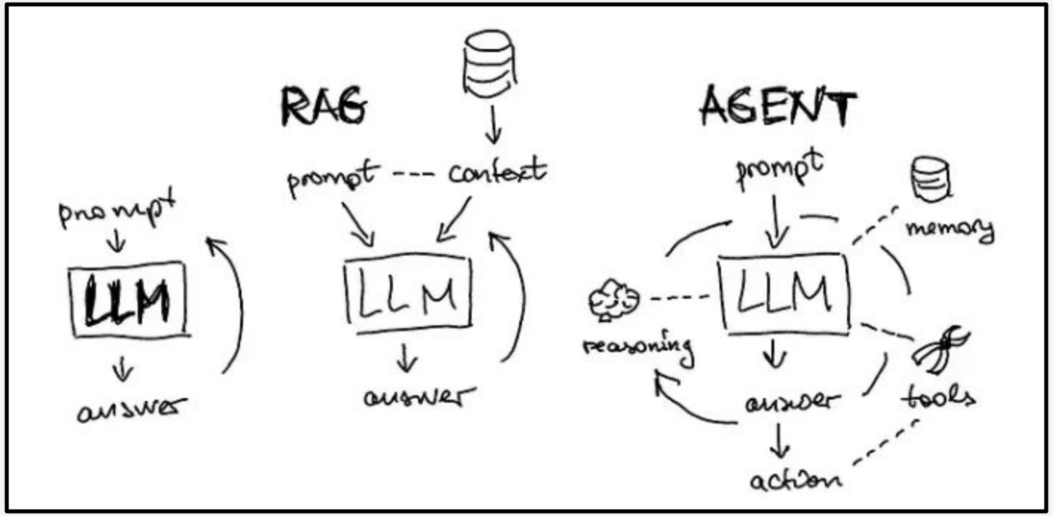
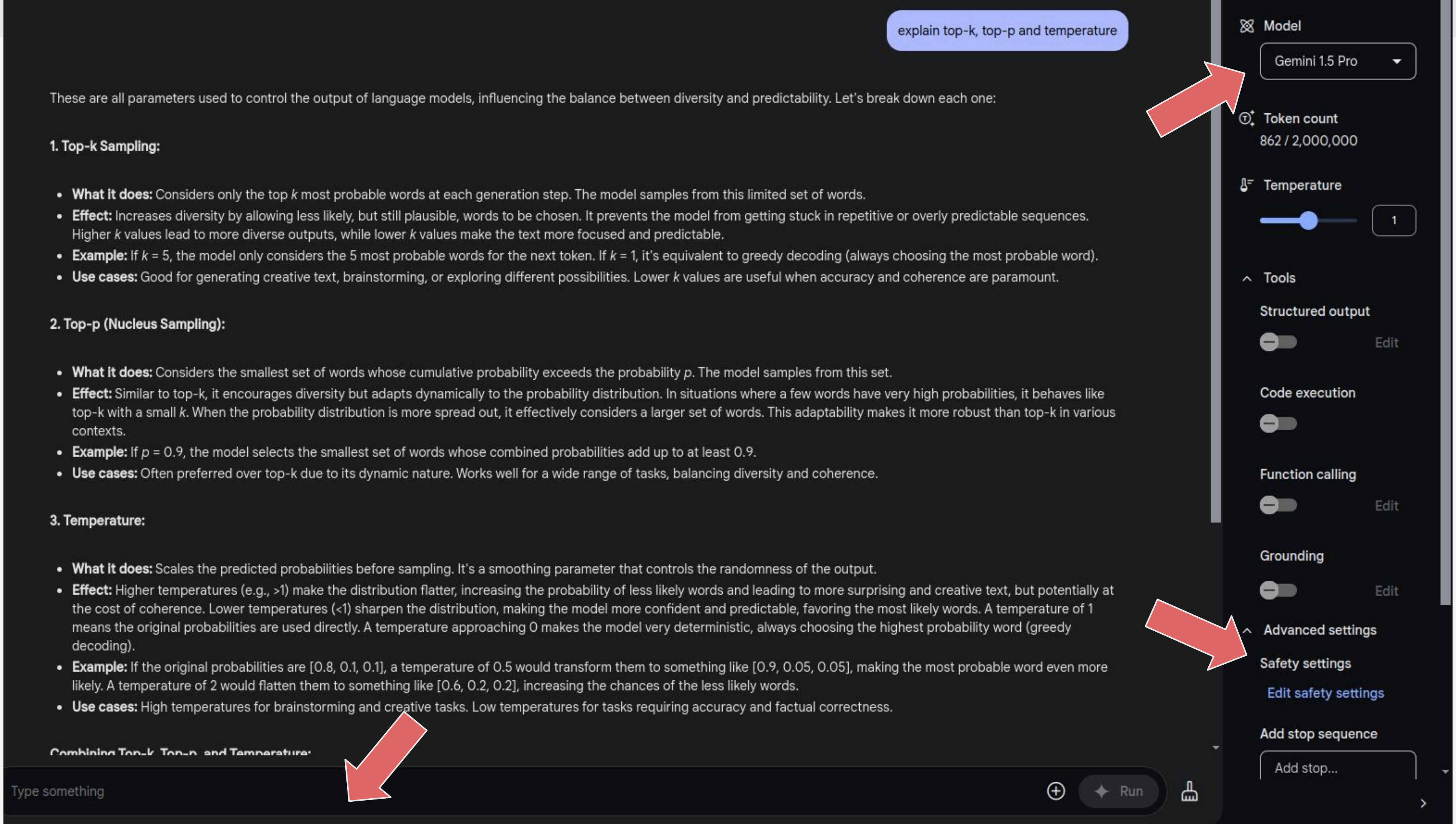
▶ Harshit Gupta Student Researcher iREL, IIIT Hyderabad 13th December 2024

LLMs and Agents: A Deep Dive Tutorial





explain top-k, top-p and temperature

These are all parameters used to control the output of language models, influencing the balance between diversity and predictability. Let's break down each one:

1. Top-k Sampling:

- **What it does:** Considers only the top *k* most probable words at each generation step. The model samples from this limited set of words.
- **Effect:** Increases diversity by allowing less likely, but still plausible, words to be chosen. It prevents the model from getting stuck in repetitive or overly predictable sequences. Higher *k* values lead to more diverse outputs, while lower *k* values make the text more focused and predictable.
- **Example:** If *k* = 5, the model only considers the 5 most probable words for the next token. If *k* = 1, it's equivalent to greedy decoding (always choosing the most probable word).
- **Use cases:** Good for generating creative text, brainstorming, or exploring different possibilities. Lower *k* values are useful when accuracy and coherence are paramount.

2. Top-p (Nucleus Sampling):

- **What it does:** Considers the smallest set of words whose cumulative probability exceeds the probability *p*. The model samples from this set.
- **Effect:** Similar to top-k, it encourages diversity but adapts dynamically to the probability distribution. In situations where a few words have very high probabilities, it behaves like top-k with a small *k*. When the probability distribution is more spread out, it effectively considers a larger set of words. This adaptability makes it more robust than top-k in various contexts.
- **Example:** If *p* = 0.9, the model selects the smallest set of words whose combined probabilities add up to at least 0.9.
- **Use cases:** Often preferred over top-k due to its dynamic nature. Works well for a wide range of tasks, balancing diversity and coherence.

3. Temperature:

- **What it does:** Scales the predicted probabilities before sampling. It's a smoothing parameter that controls the randomness of the output.
- **Effect:** Higher temperatures (e.g., >1) make the distribution flatter, increasing the probability of less likely words and leading to more surprising and creative text, but potentially at the cost of coherence. Lower temperatures (<1) sharpen the distribution, making the model more confident and predictable, favoring the most likely words. A temperature of 1 means the original probabilities are used directly. A temperature approaching 0 makes the model very deterministic, always choosing the highest probability word (greedy decoding).
- **Example:** If the original probabilities are [0.8, 0.1, 0.1], a temperature of 0.5 would transform them to something like [0.9, 0.05, 0.05], making the most probable word even more likely. A temperature of 2 would flatten them to something like [0.6, 0.2, 0.2], increasing the chances of the less likely words.
- **Use cases:** High temperatures for brainstorming and creative tasks. Low temperatures for tasks requiring accuracy and factual correctness.

Combining Top-k, Top-p, and Temperature:

Model

Gemini 1.5 Pro

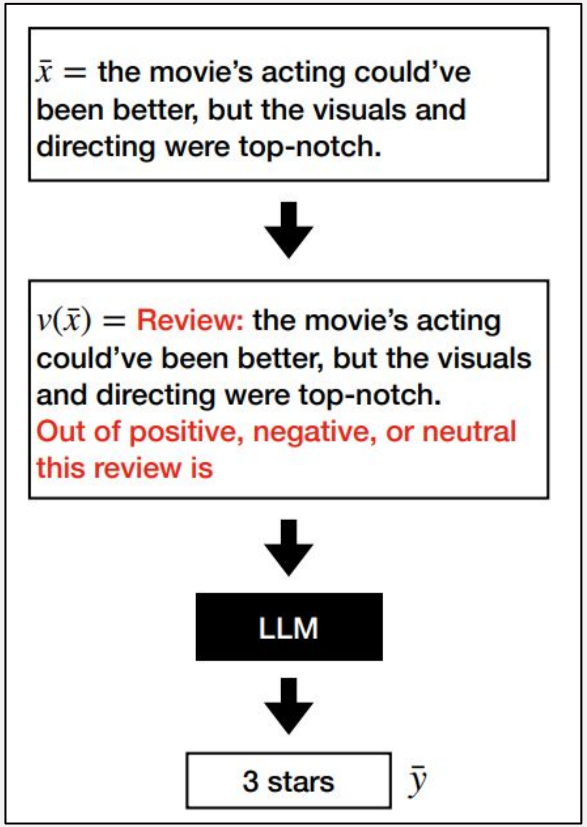
Token count

862 / 2,000,000

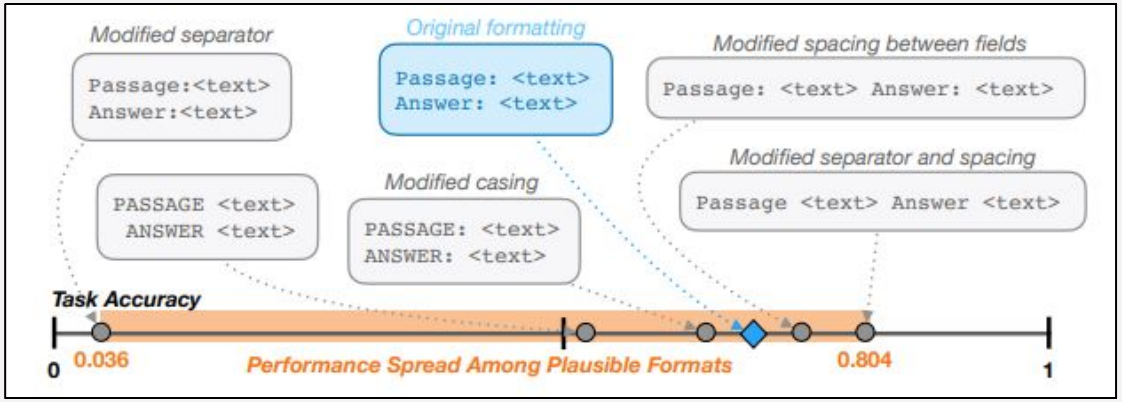
Temperature

1

Zero-Shot Prompting

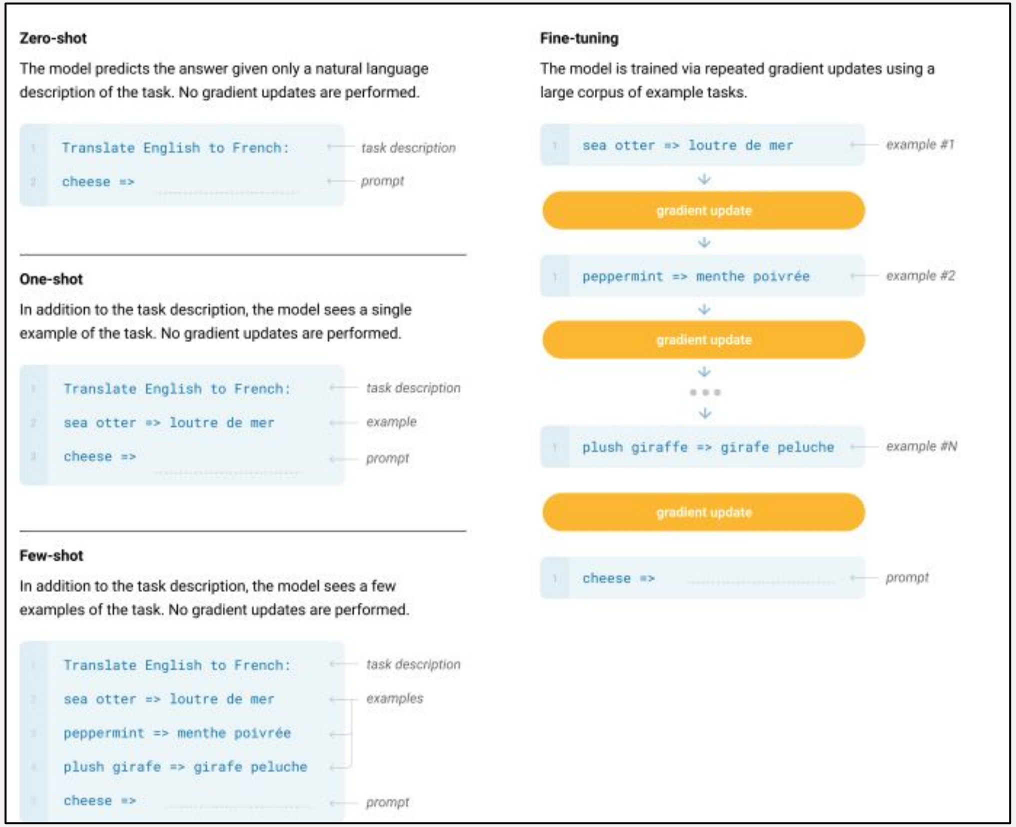


$\bar{x}$ = the movie's acting could've been better, but the visuals and directing were top-notch.
$v(\bar{x})$ = Review: the movie's acting could've been better, but the visuals and directing were top-notch. Out of positive, negative, or neutral this review is
LLM
3 stars
$\bar{y}$



1. Prompts can even be sensitive to minor cosmetic changes.
2. Can influence performance in unexpected ways.
3. Can think of them as (very complex) hyper-parameters.

In-Context Learning (ICL)



Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:     ← task description
cheese =>                        ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:     ← task description
sea otter => loutre de mer       ← example
cheese =>                        ← prompt
```

Few-shot

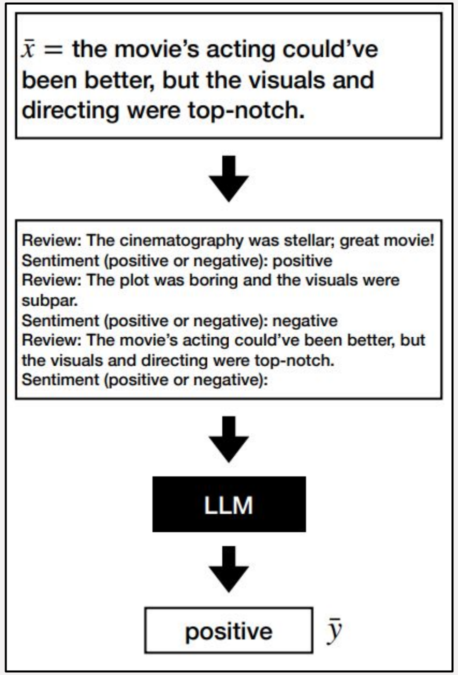
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:     ← task description
sea otter => loutre de mer       ← examples
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>                        ← prompt
```

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

```
sea otter => loutre de mer       ← example #1
gradient update
peppermint => menthe poivrée     ← example #2
gradient update
• • •
plush giraffe => girafe peluche  ← example #N
gradient update
cheese =>                        ← prompt
```



$\bar{x}$ = the movie's acting could've been better, but the visuals and directing were top-notch.

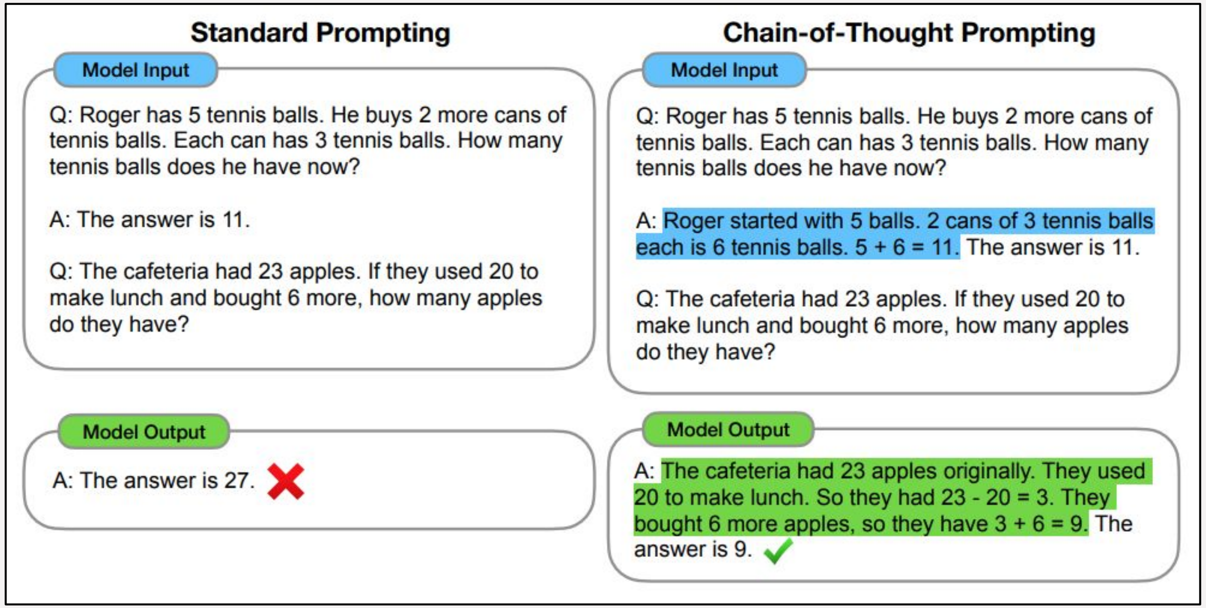
Review: The cinematography was stellar; great movie!
Sentiment (positive or negative): positive
Review: The plot was boring and the visuals were subpar.
Sentiment (positive or negative): negative
Review: The movie's acting could've been better, but the visuals and directing were top-notch.
Sentiment (positive or negative):

ICL can be highly sensitive to the choice of examples, their ordering, and the format of the prompt.

Chain-Of-Thought (COT) Prompting

Directly generating the answer (shortcut)

Show the model examples of the reasoning steps through ICL



Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

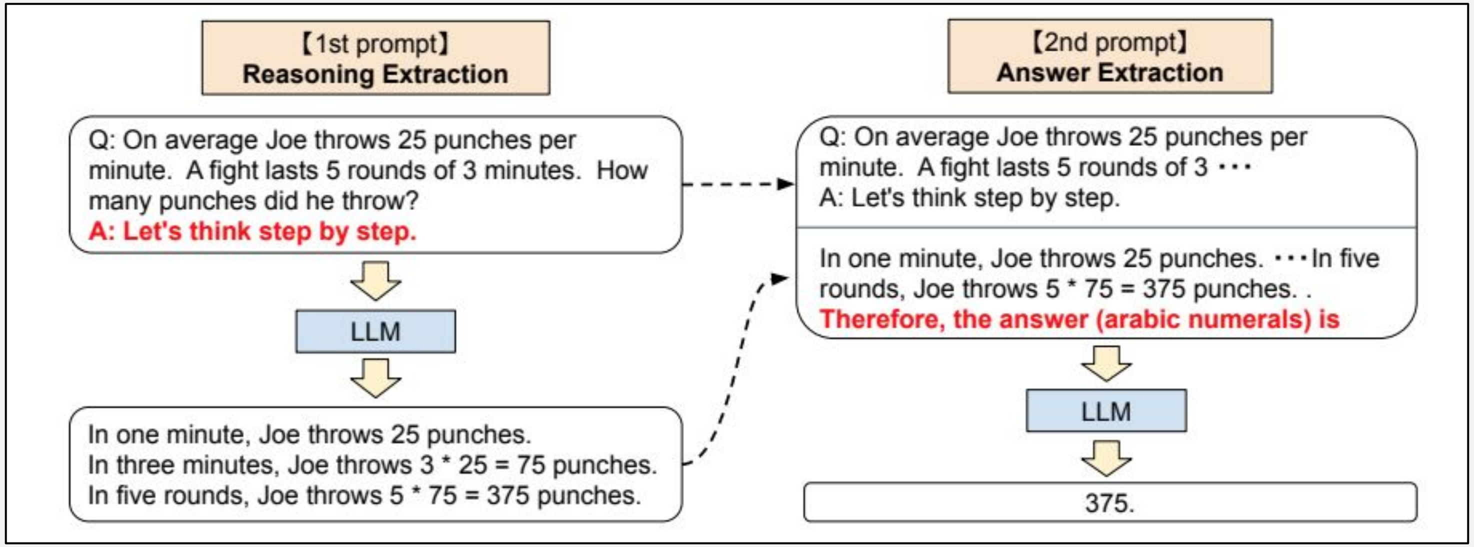
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

And then have it explicitly generate the reasoning steps


【1st prompt】
Reasoning Extraction
Q: On average Joe throws 25 punches per minute. A fight lasts 5 rounds of 3 minutes. How many punches did he throw?
A: Let's think step by step.
LLM
In one minute, Joe throws 25 punches.
In three minutes, Joe throws 3 * 25 = 75 punches.
In five rounds, Joe throws 5 * 75 = 375 punches.
【2nd prompt】
Answer Extraction
Q: On average Joe throws 25 punches per minute. A fight lasts 5 rounds of 3 ···
A: Let's think step by step.
In one minute, Joe throws 25 punches. ··· In five rounds, Joe throws 5 * 75 = 375 punches. .
Therefore, the answer (arabic numerals) is
LLM
375.

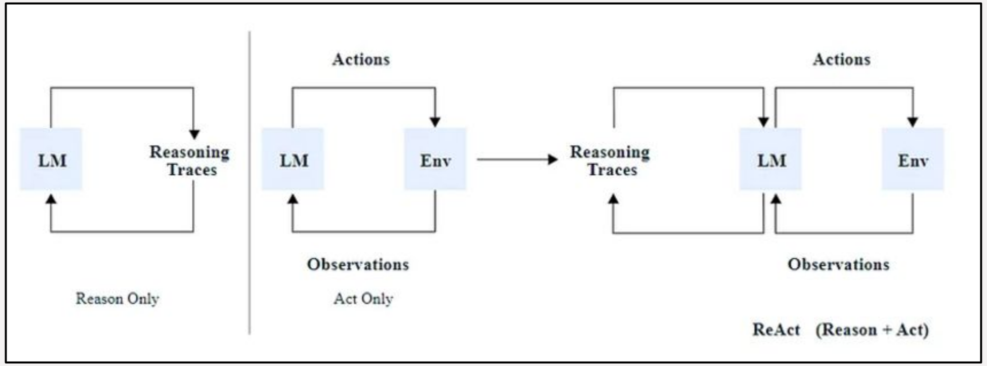
Challenge: the answer is often entangled in the reasoning text — how to extract it? → just use an LLM

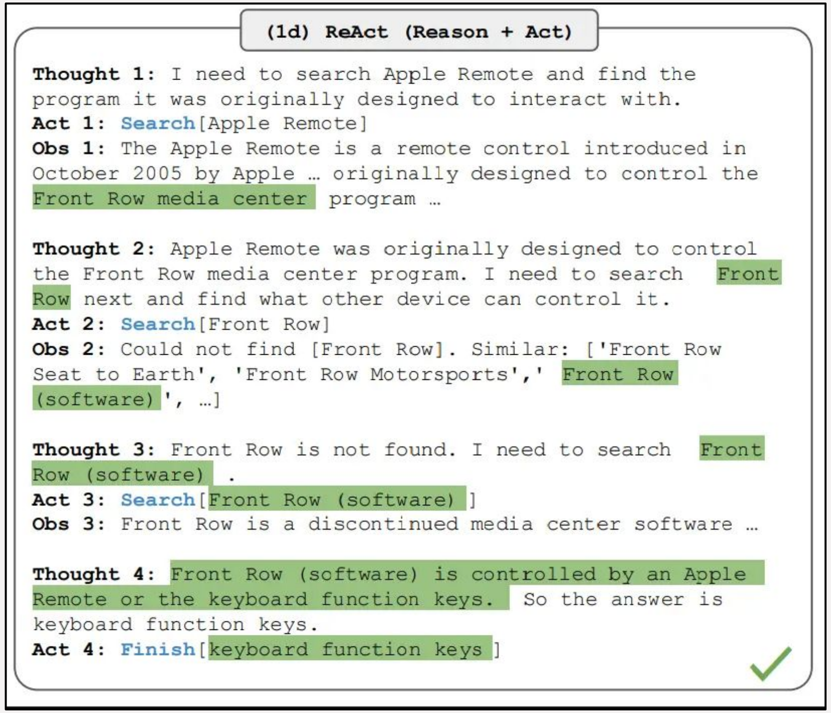
Main Idea: Just “tell” the model to reason in steps

THE PROMPTING TECHNIQUES DEMO

ReAct Prompting : ReAct = Reason + Act with LLMs

ReAct is inspired by the synergies between **"acting"** and **"reasoning"** which allow humans to learn new tasks and make decisions or reasoning.





(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

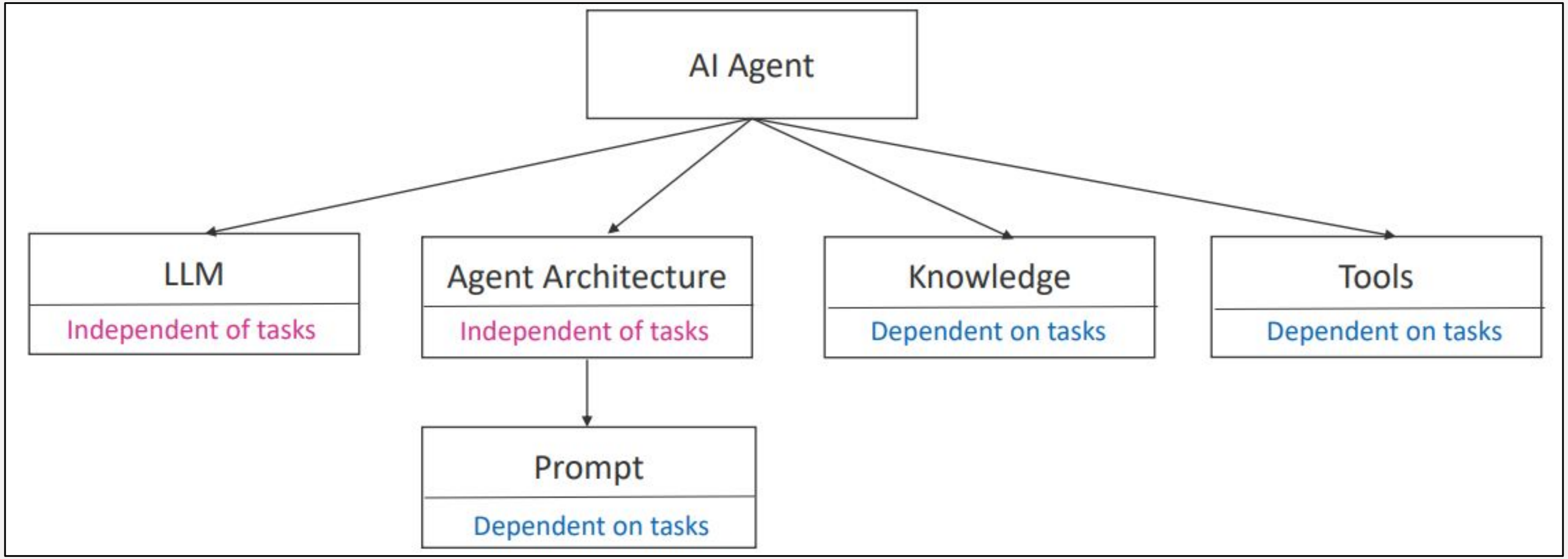
Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys]

✓

THE BENEFITS OF REACT PROMPTING DEMO

AI Agents from a Developer's Perspective

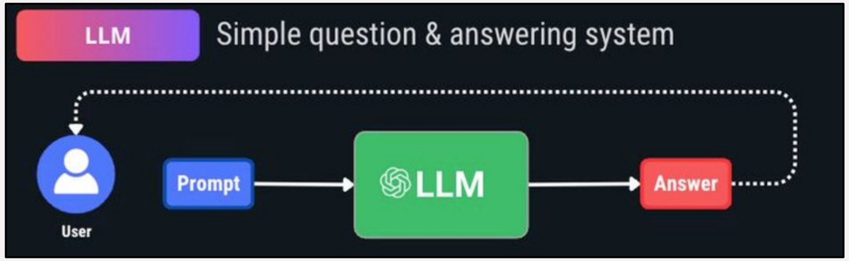
- AI agents can be divided into task-dependent and task-independent parts.
- By tuning the parts that depend on tasks, they can be specialized for specific tasks.
- LLMs and Agent Architectures are designed to be versatile and independent of specific tasks.



LLM and RAG are not Agents

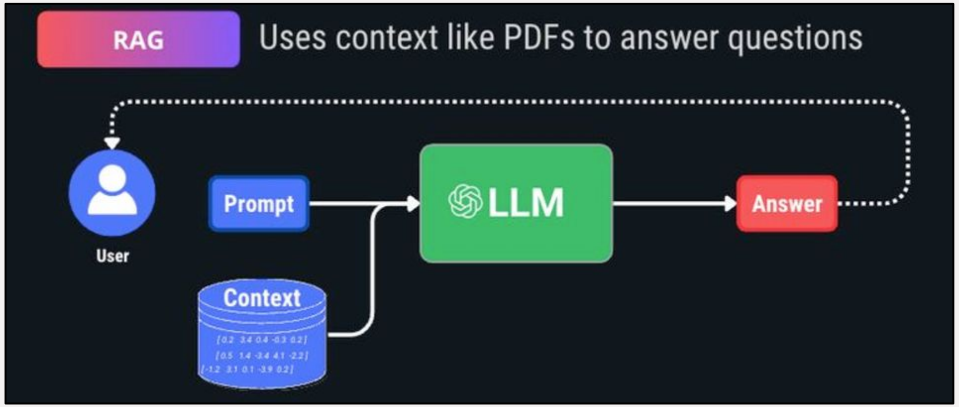
LLM: Generates text based on prompts

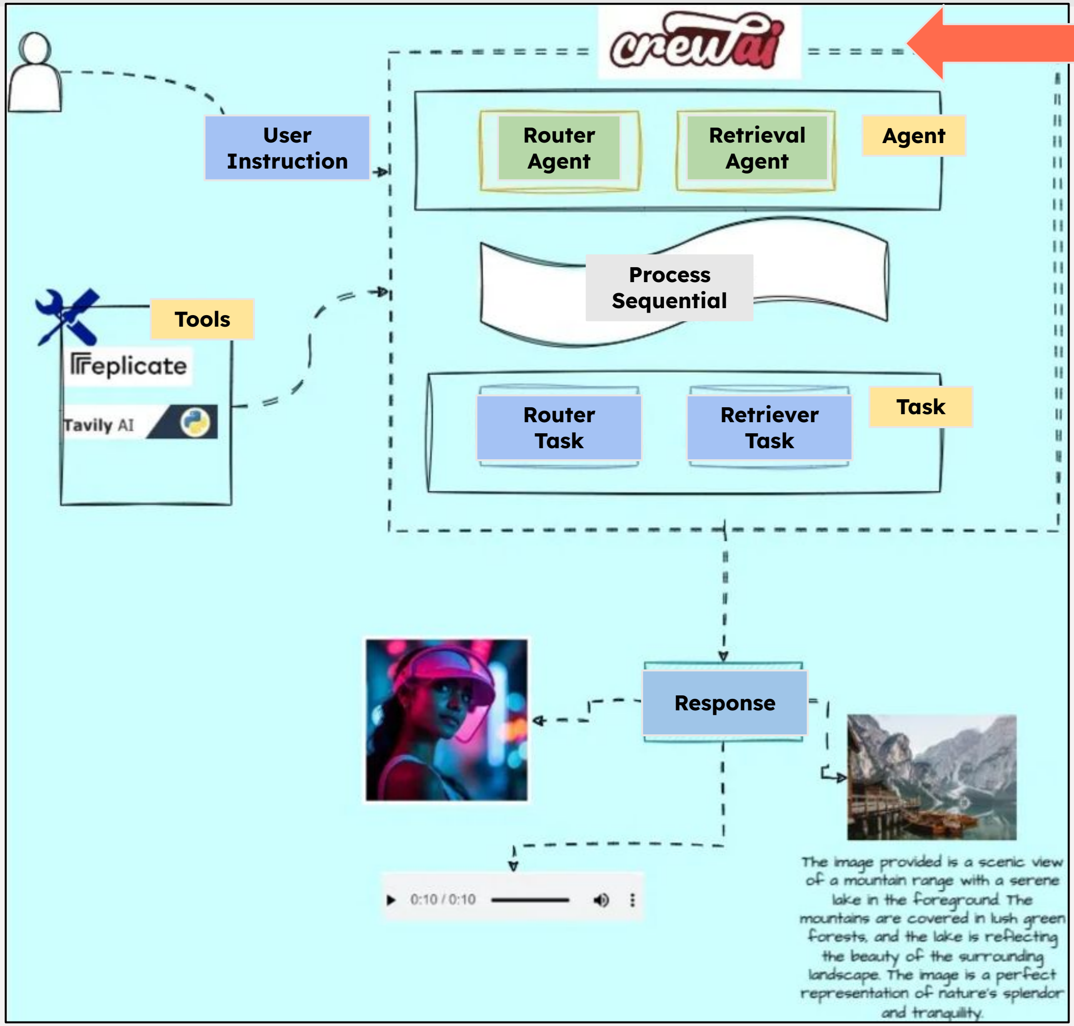
1. Cannot execute actions
2. Cannot retrieve environment information



RAG: Generates text by passing document search results as text prompts

1. Cannot continue searching until the goal is achieved
2. Cannot adapt search content based on search results to decide the next search
3. Requires customization from search to prompt entry for each purpose



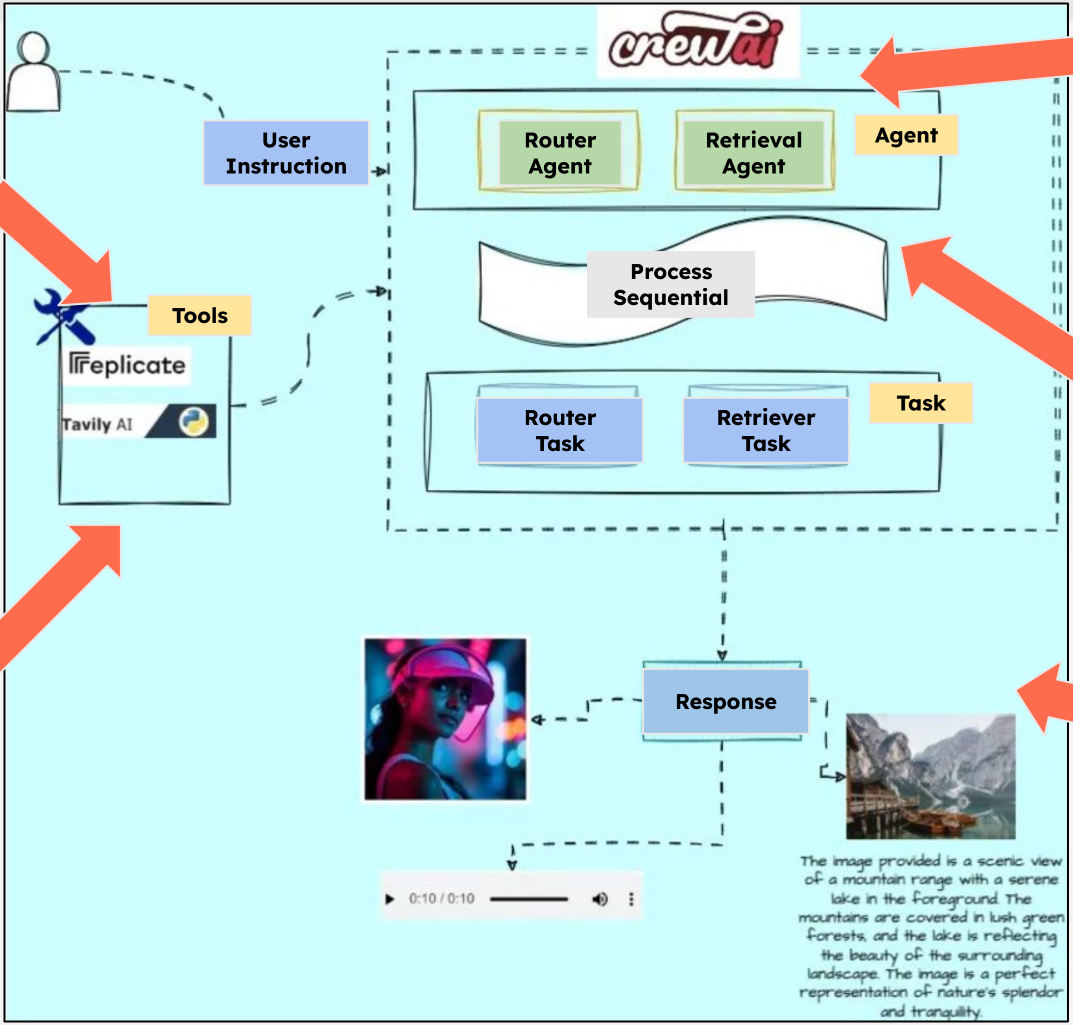

crewai
User Instruction
Router Agent
Retrieval Agent
Agent
Tools
replicate
Tavily AI
Process Sequential
Router Task
Retriever Task
Task
Response
0:10 / 0:10
The image provided is a scenic view of a mountain range with a serene lake in the foreground. The mountains are covered in lush green forests, and the lake is reflecting the beauty of the surrounding landscape. The image is a perfect representation of nature's splendor and tranquility.

CrewAI allows developers to build complex, collaborative workflows where agents assume specific roles and work together to achieve common objectives.

CrewAI introduced Flows. Flows provide a structured, event-driven framework for creating and managing AI workflows, allowing developers to seamlessly connect multiple tasks, manage state, and control execution flow within AI applications.

The agent can select from a curated set of tools to accomplish specific tasks, similar to the functionality offered by LangChain agents.

Tavily-Python: Open source library used for web searching and information retrieval.



The image provided is a scenic view of a mountain range with a serene lake in the foreground. The mountains are covered in lush green forests, and the lake is reflecting the beauty of the surrounding landscape. The image is a perfect representation of nature's splendor and tranquility.

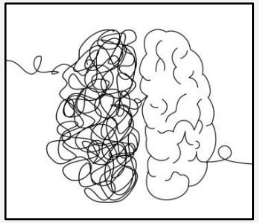
Based on the user Instruct the Router Agent decides on further course of action.

Based on the response from the Router Agent the Retriever Agents performs the final task by invoking the respective tool.

The agent can handle various modalities, enabling it to perform tasks across text, images, and audio.

AGENTS DEMO

Potential Downsides



Increased Complexity
Managing multiple microservices can be complex, especially when coordinating interactions between various AI agents. This complexity can lead to challenges in debugging and maintaining the system.

Latency Issues
Communication between microservices can introduce latency, which might affect the performance of AI agents that require real-time processing.

Resource Overhead
Each microservice typically runs in its own container or virtual machine, which can lead to higher resource consumption compared to a monolithic architecture.

Data Consistency
Ensuring data consistency across distributed microservices can be challenging, especially when dealing with stateful AI agents.

Security Concerns
Each microservice can be a potential attack vector, increasing the overall security risk. Proper security measures need to be implemented to protect the system.

Thank you