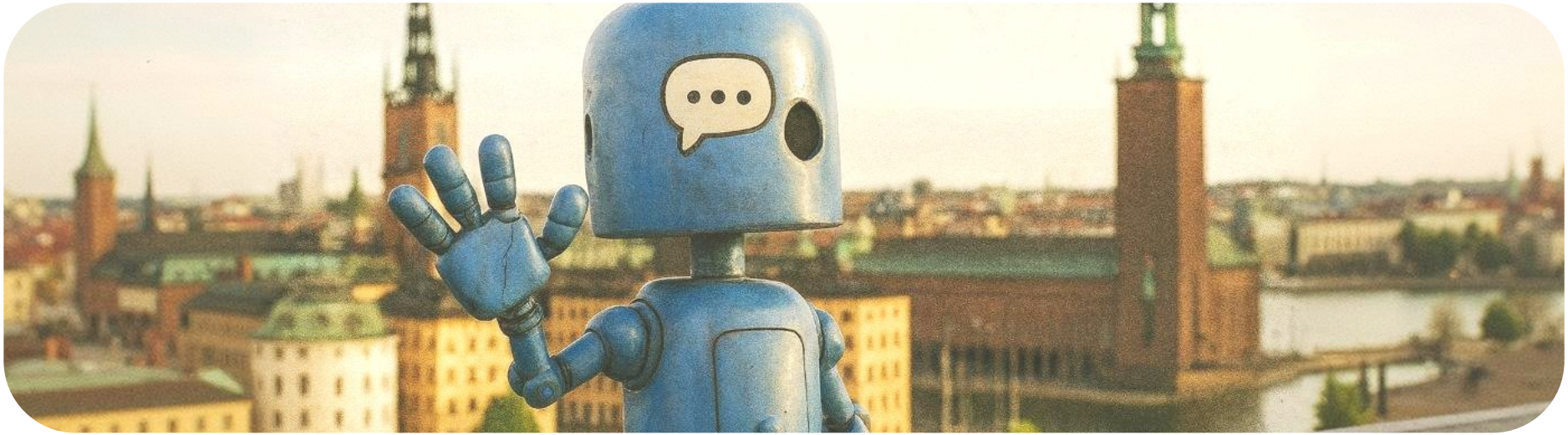


Practical Things about Productizing LLM Agent System

Lele Cao

Thanks to

About Lele

Senior Principal AI/ML Researcher
Research Lead
@ King/Microsoft

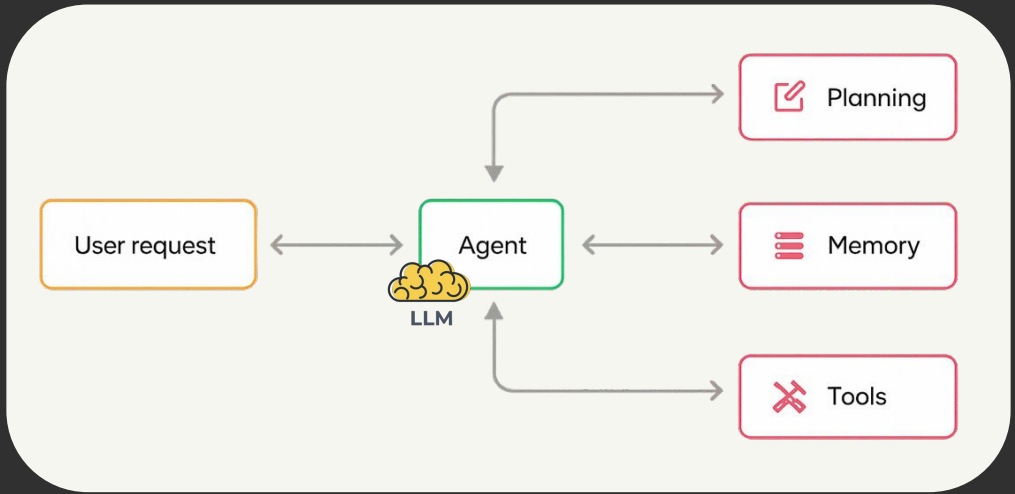
Industrial Part

- 16-year in King/Microsoft, EQT Motherbrain, Alibaba, Elisa and Ericsson.
- AI/ML Engineer, Data Scientist, Research Engineer, Software Engineer, Product Designer, Engineering Manager, General Manager.

Academic Part

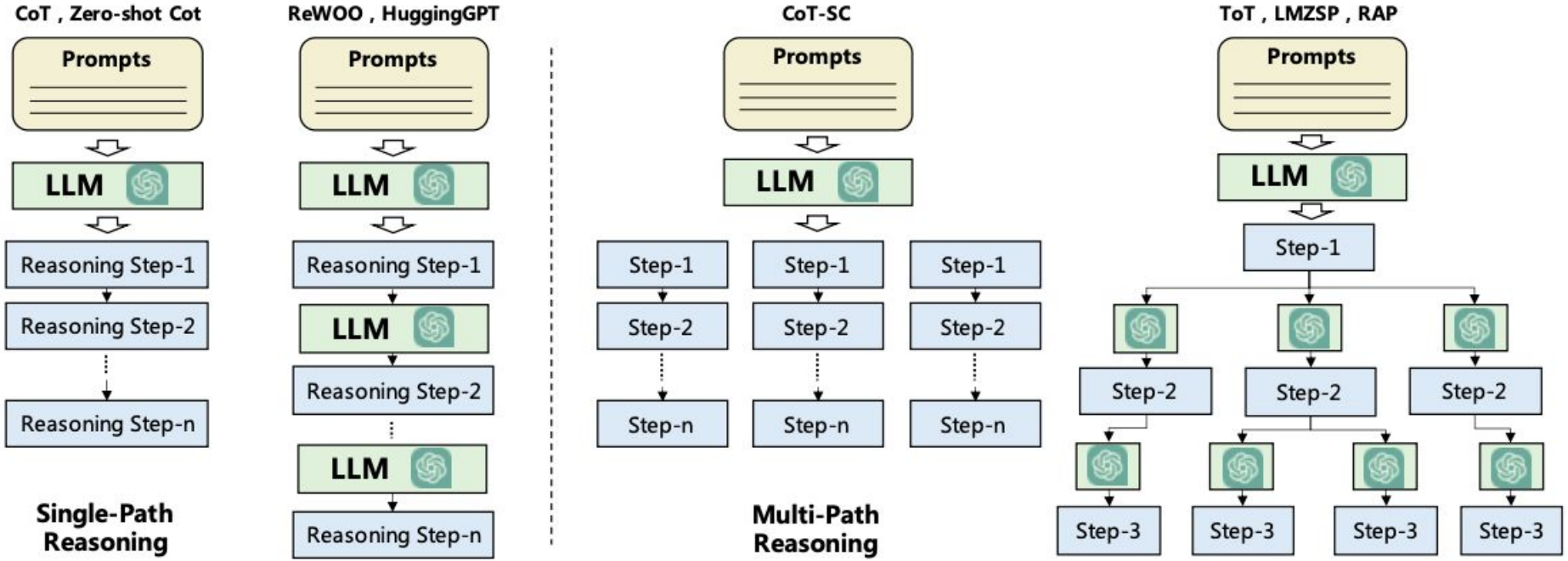
- 4-year BS in Software Engineering
- 2-year MS in Interactive Systems Design
- 4-year PhD in AI and Robotics
- 2016: Almost started a Professorship career.
- >50 publications on deep learning, graph learning, timeseries, language model, robotics, reinforcement learning, computer vision, explainability ...

LLM Agent



A system powered by a Large Language Model (LLM) that goes beyond simple text generation to perform **goal-oriented** and **multi-step** tasks by planning, reasoning & interacting with tools and/or data.

Planning & Reasoning



Wang, Lei, et al. "A survey on large language model based autonomous agents." Frontiers of Computer Science 18.6 (2024): 186345.

"When the steps are predefined (*not determined by LLMs on the fly*), it's an Agent Workflow."

Lele Cao

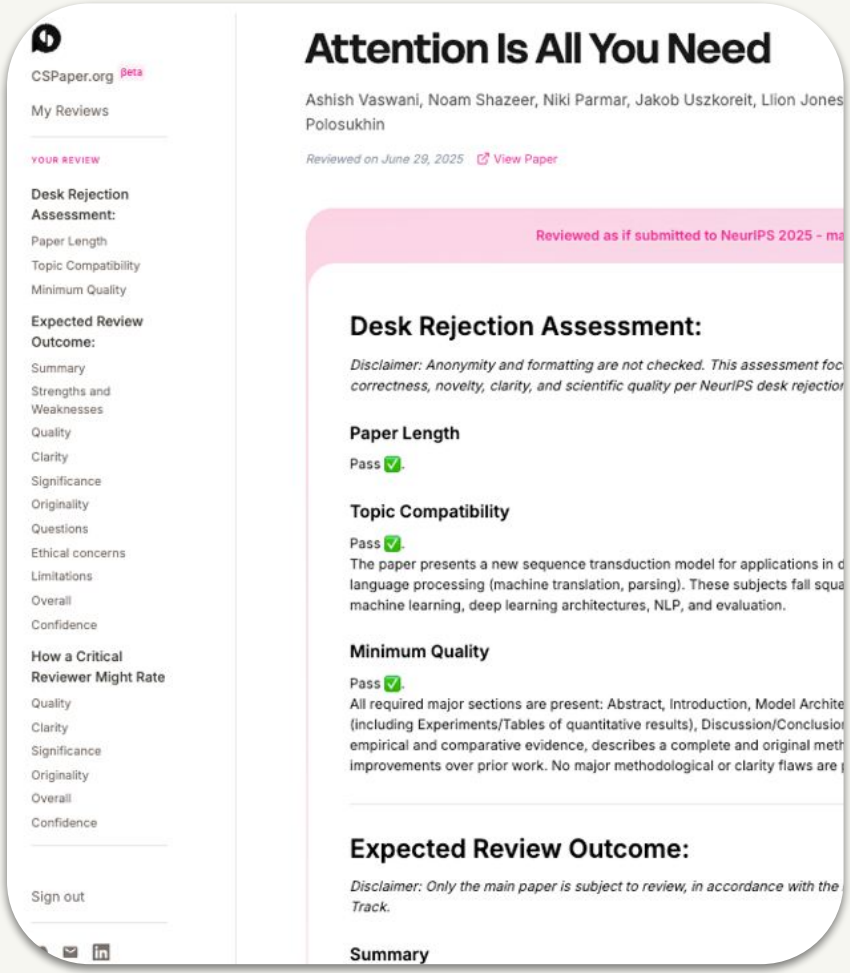
Where

my learnings come from

https://cspaper.org

AI agent system to help researchers to obtain fast and conference-specific review for their research papers.

A Short Paper:
CSPaper Review: Fast, Rubric-Faithful Conference Feedback



Practical Things

Start with stateless agent workflow

Tools unlock true LLM-agnostic system

Code first. LLM when necessary

Optimize task density per token

Affordance: not every app should be a chat

How does a stateful multi-round chat remember things?

A

The LLM itself is updated with all historical information gradually.

B

Things are distilled into compact knowledge-base and becomes a part of LLM.

C

Things are stored in DB as is, and LLM knows how to pick relevant info.

D

All historical things always are sent as they are cumulatively to new chat rounds.

Start with stateless agent workflow

How does a stateful multi-round chat remember things?

A

The LLM itself is updated with all historical information gradually.

B

Things are distilled into compact knowledge-base and becomes a part of LLM.

C

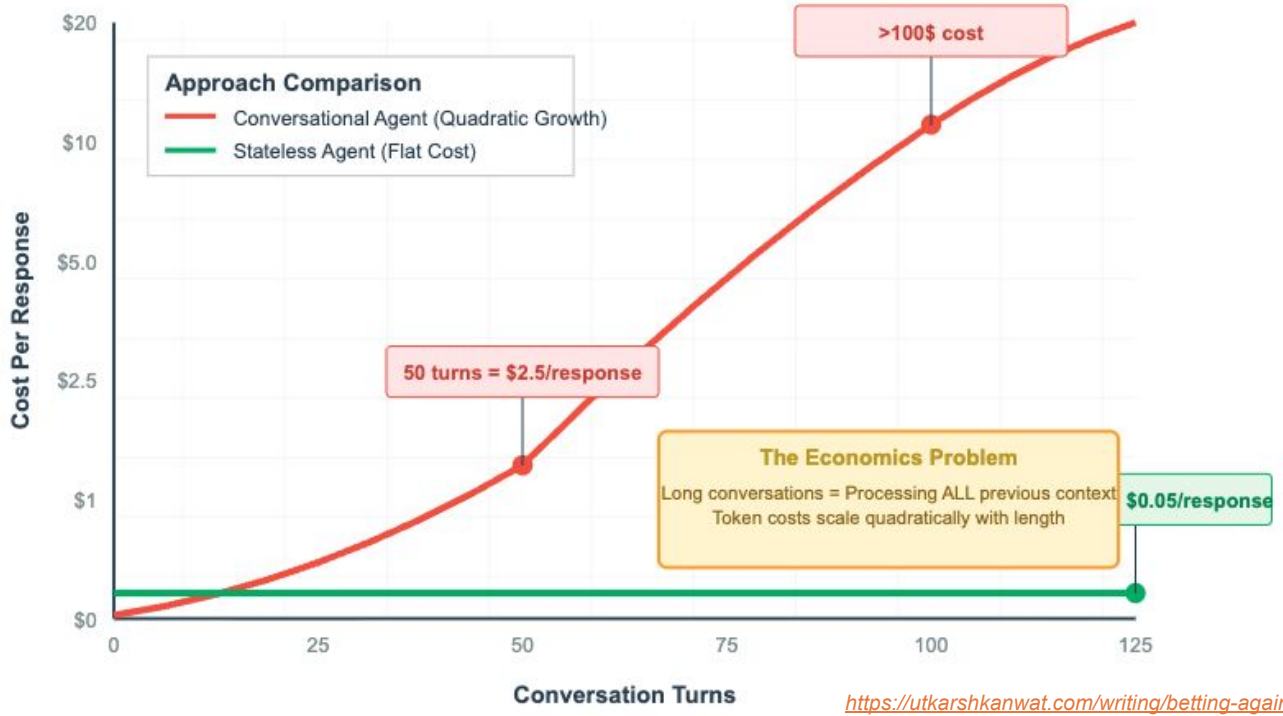
Things are stored in DB as is, and LLM knows how to pick relevant info.

D

All historical things always are sent as they are cumulatively to new chat rounds.

Start with stateless agent workflow

Non-sustainable token economics

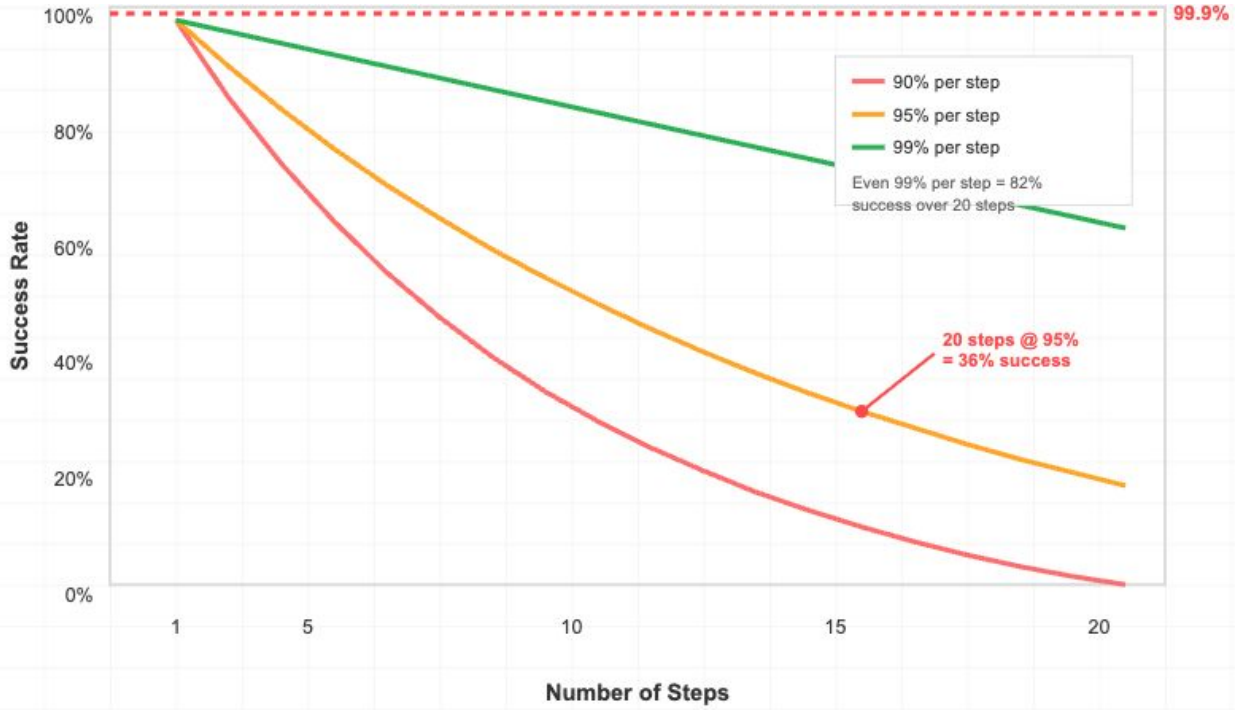


https://utkarshkanwat.com/writing/betting-against-agents

Start with stateless agent workflow

Errors compound exponentially

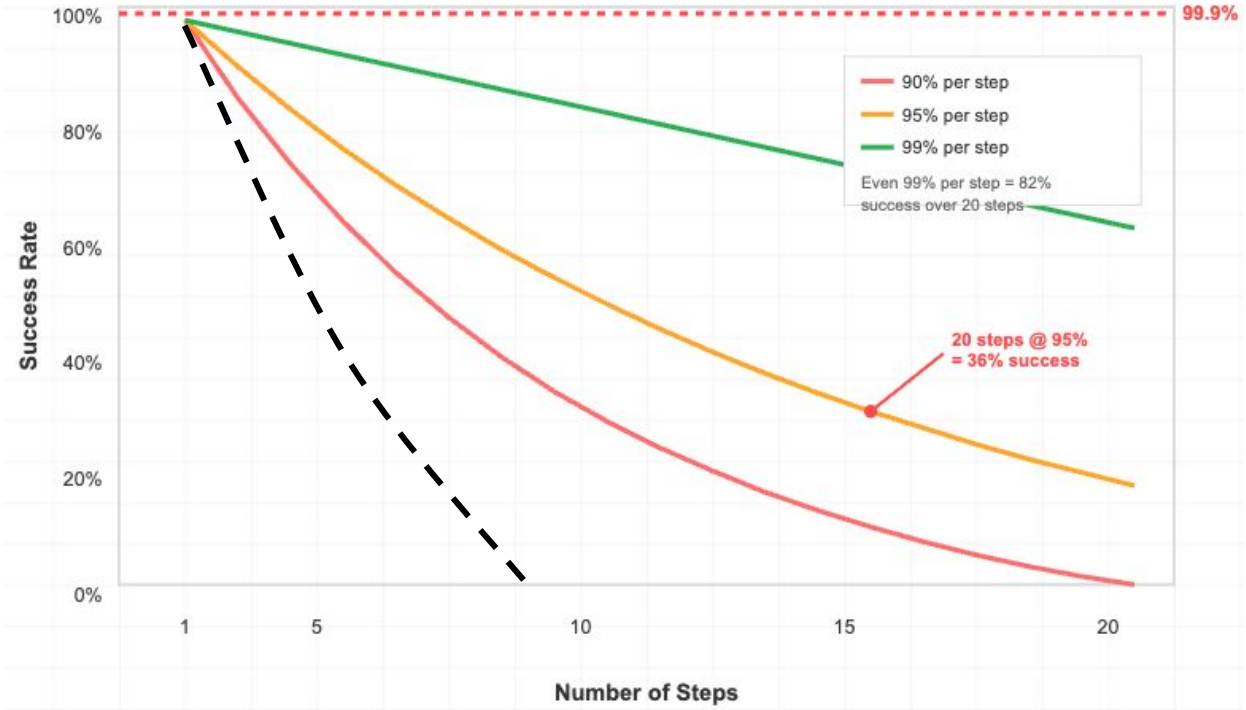
If you must use multi-step agent, make sure you **capture** (be it automatically or manually) and **quarantine** errors in each step!



Start with stateless agent workflow

Long Context ≠ Good Quality

- Poisoning
- Distraction
- Confusion



Start with stateless agent workflow

Implications

- Minimize the times/steps you need LLM, preferably only once.
- Use agent workflow, especially when you know the problem domain inside-out.
- Make your agent/steps stateless as much as possible.
- Only provide your agent directly relevant and verified information.
- Ofc, you can mix-and-match!

Start with stateless agent workflow

Practical Things

Start with stateless agent workflow

Tools unlock true LLM-agnostic system

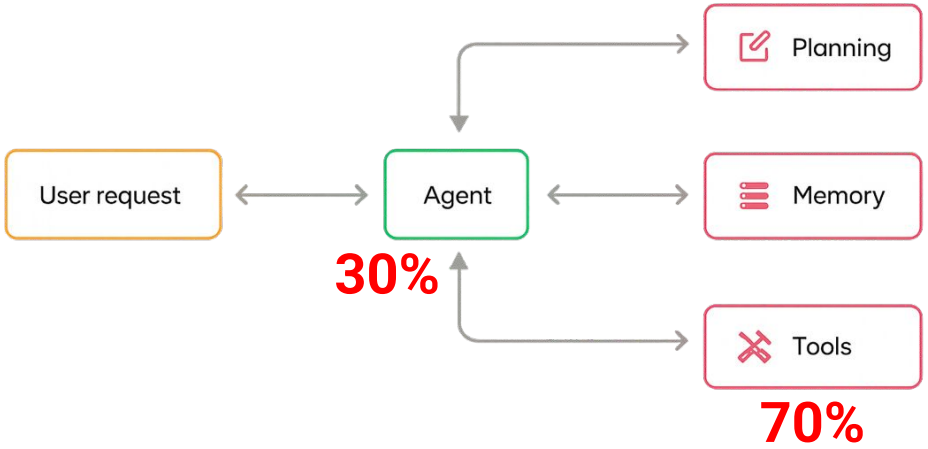
Code first. LLM when necessary

Optimize task density per token

Affordance: not every app should be a chat

The success mostly on tools

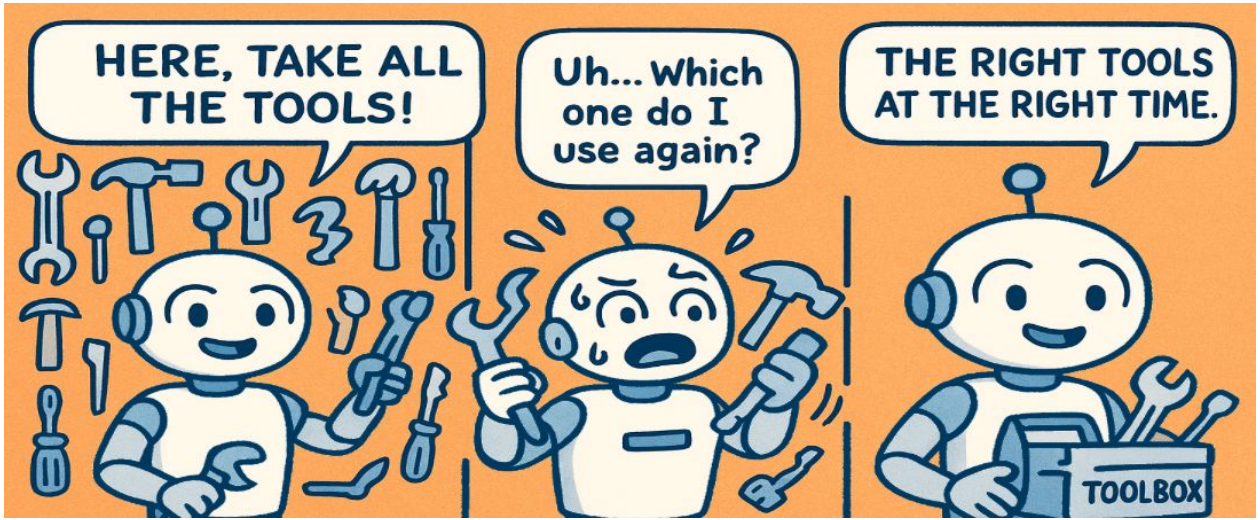
- With input from great tools, you can choose pretty much any established LLMs without significant performance decay.



Tools unlock true LLM-agnostic system

But, tool overload is also real

- More tools in the prompt → worse tool selection. Small models call irrelevant tools more often.
- Dynamic tool loading: rank/select tools per turn (e.g., semantic lookup over tool descriptions)



Tools unlock true LLM-agnostic system

Practical Things

Start with stateless agent workflow

Tools unlock true LLM-agnostic system

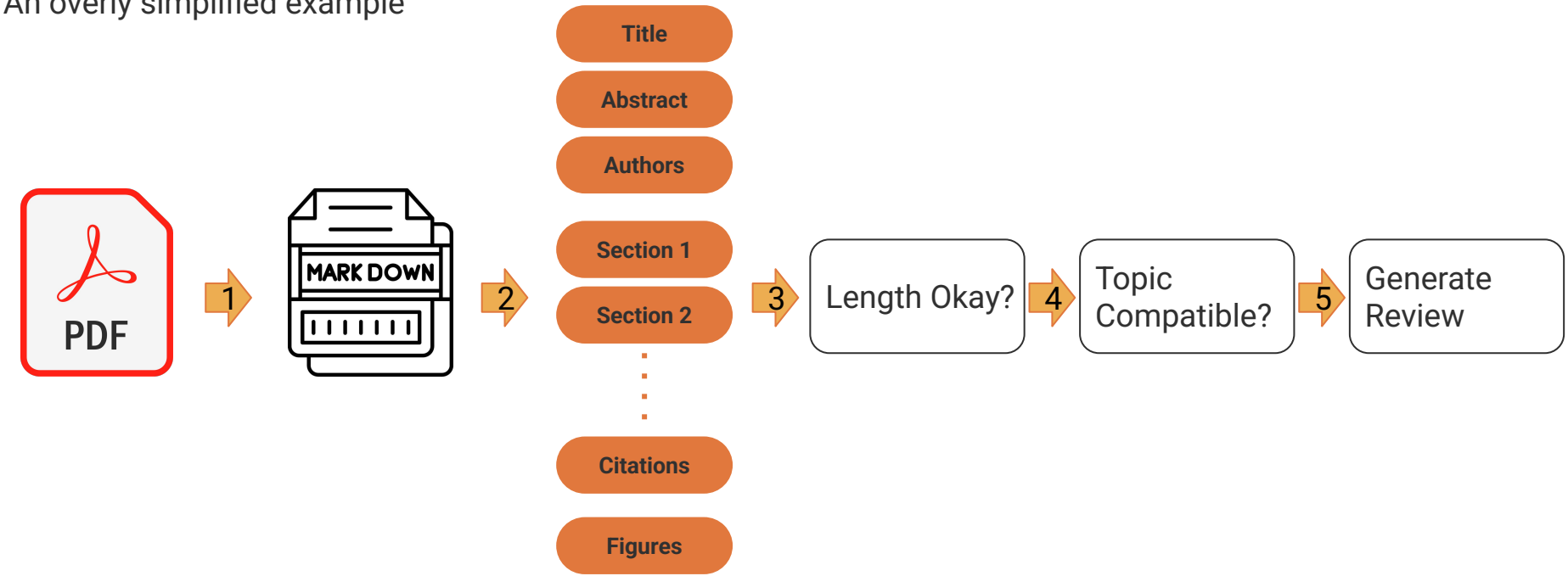
Code first. LLM when necessary

Optimize task density per token

Affordance: not every app should be a chat

Which steps have to use LLM?

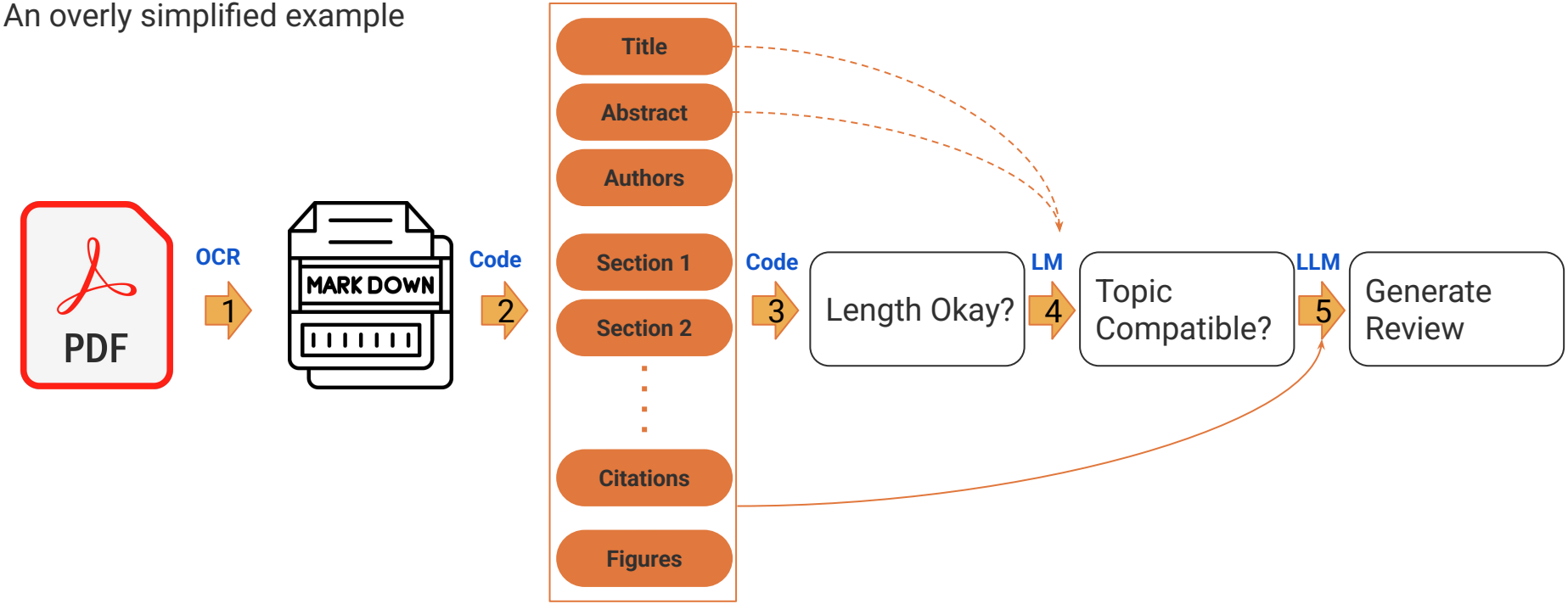
An overly simplified example



Code first. LLM when necessary

Which steps have to use LLM?

An overly simplified example



Code first. LLM when necessary

Practical Things

Start with stateless agent workflow

Tools unlock true LLM-agnostic system

Code first. LLM when necessary

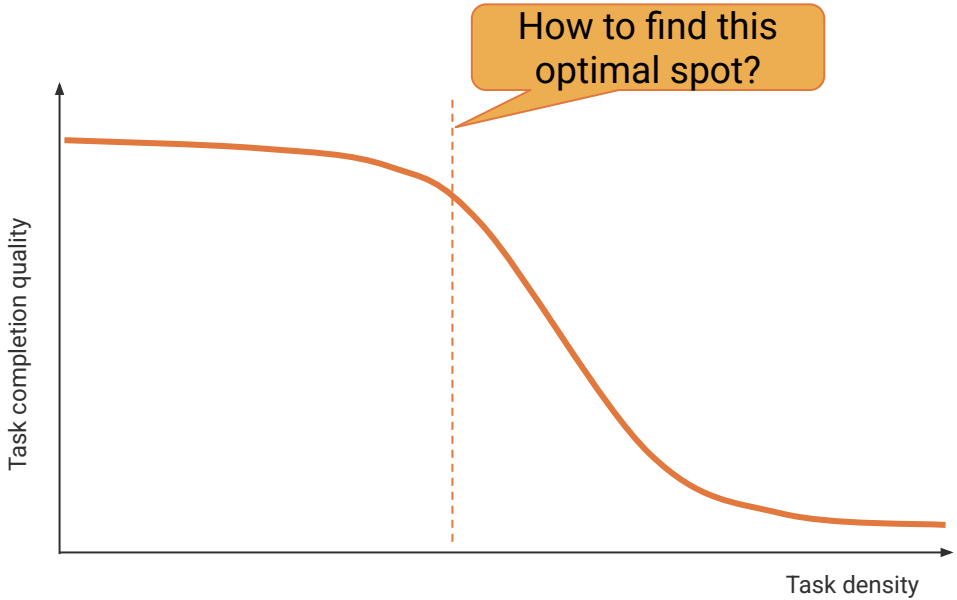
Optimize task density per token

Affordance: not every app should be a chat

Task density vs. completion quality

Task density measures how much intended reasoning, problem-solving, or challenge is packed into each unit of computational resource (e.g., tokens consumed).

$$\textbf{TaskDensity} = \frac{\text{Task Challengeness or Cognitive Load}}{\text{Tokens Consumed}}$$



Optimize task density per token

Benchmarking – an example

Generate Review

A. Trigger one LLM call for each paper section
B. Call LLM only once providing all sections

Task Density	Task completion quality (MAE)	# Tokens per paper
A (low)	0.12	~100k
B (high)	0.15	~10k

Optimize task density per token

Practical Things

Start with stateless agent workflow

Tools unlock true LLM-agnostic system

Code first. LLM when necessary

Optimize task density per token

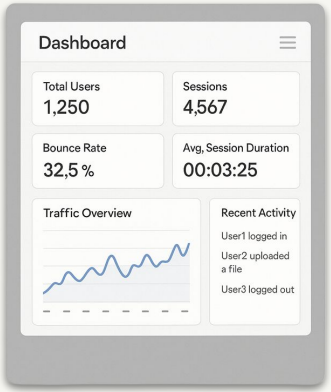
Affordance: not every app should be a chat

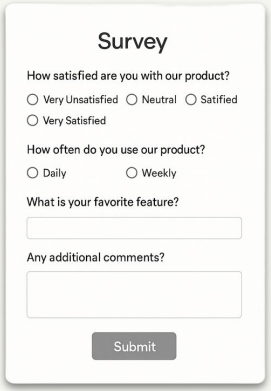
Don’t craft a hammer without a nail

- Make your design naturally imply its intended usage – minimize the need of educating your user or forcing them read manual!
- Support safe exploration – free chat is not such an environment most of the time!
- Get a domain expert!
- Get an interaction designer!
- Deliver swiftly and minimally while iterating with your users!

Affordance: not every app should be a chat

Final Thoughts





Do not intend to get everything right the first a few iterations!

Prioritize simplicity, control, and robustness over flashiness!

Carefully track cost and usage — especially important without major investors funding you.

Thank You