From LLMs to Agents

Andrea Gemelli, PhD - 14.05.2026

Nice to meet you! 🤗

- **Master Degree of Computer Science** @ UniFi
- **European PhD** @ CVC (Barcelona) and UniFi
- **AI Research Scientist** in Paris 🇫🇷

Contacts:

Email: gemelli.andrea@hotmail.com

Portfolio/Blog: https://www.andreagemelli.me/

Cool Projects and Collaborations with Students

- **BoundingDocs** w/ Simone G.: a dataset for grounding question answering on documents
    - Dataset downloaded 20k+ on hugging face

- **DocExplainer** w/ Alessio C.: a wrapper explainable model that can be used together with “any” LLM
    - Paper https://arxiv.org/pdf/2509.10129

- What next?

Roadmap

01 **LLMs**

02 **Agents**

03 **Agentic Document AI**

PART 01

LLMs

The language unit measure: a token

LLMs are fed by text chunks called **tokens**.

For simplicity, you can think 1 token ~ 1 word (in reality 1 word can be 1 to 3 token)

Tokens are sub-word units by a trained algorithm e.g. BPE or SentencePiece.

The vocabulary is the dimension of the dictionary that stores indices (keys) to their token (chunk of text) and viceversa

Large Language Models

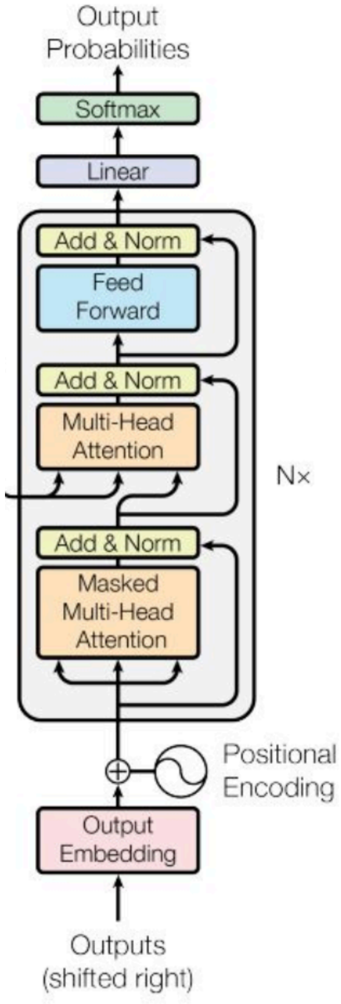
gpt-4o

Token count
15

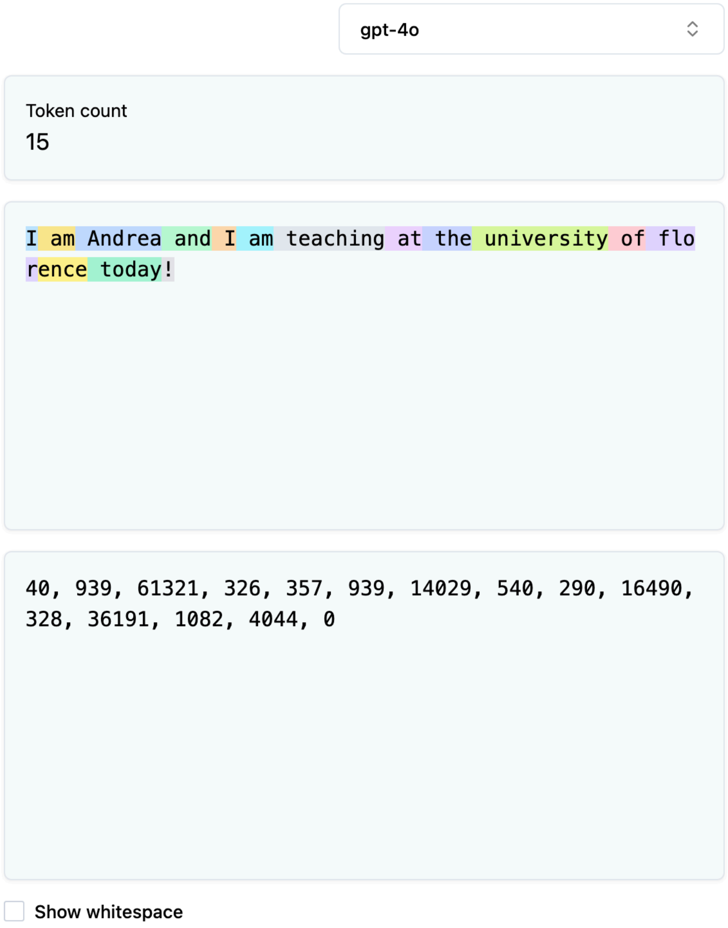
I am Andrea and I am teaching at the university of flo
rence today!

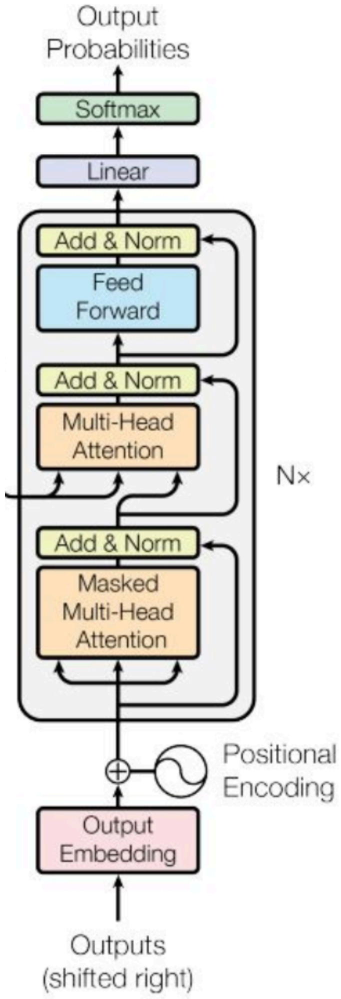
40, 939, 61321, 326, 357, 939, 14029, 540, 290, 16490,
328, 36191, 1082, 4044, 0

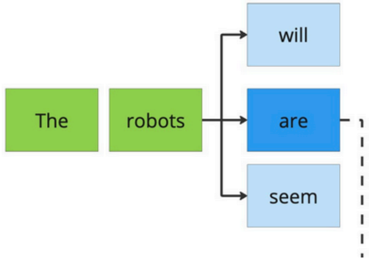
Show whitespace



Large Language Models

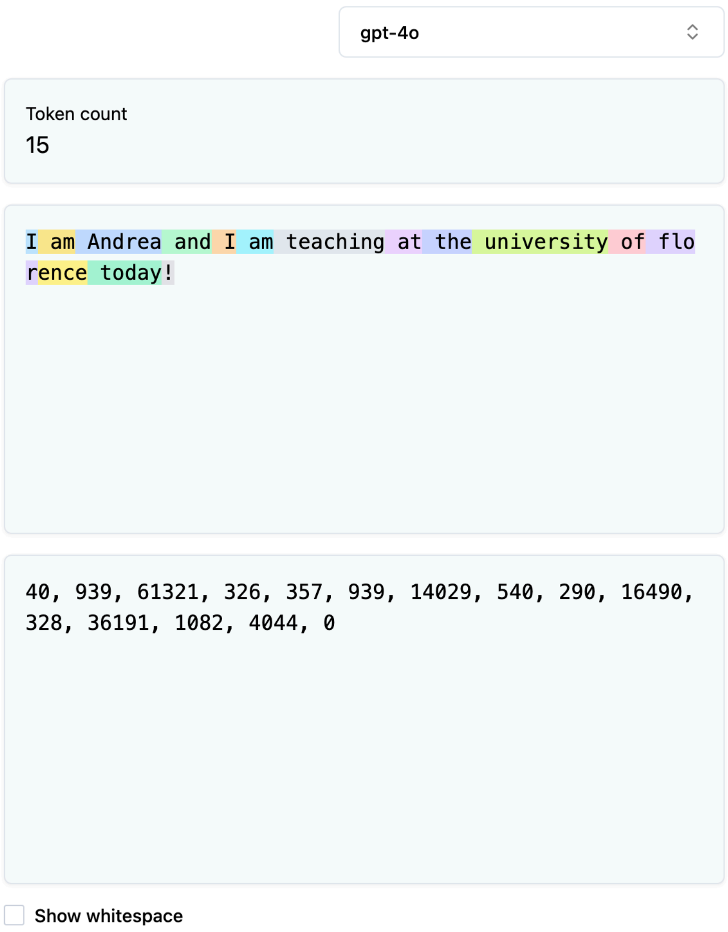


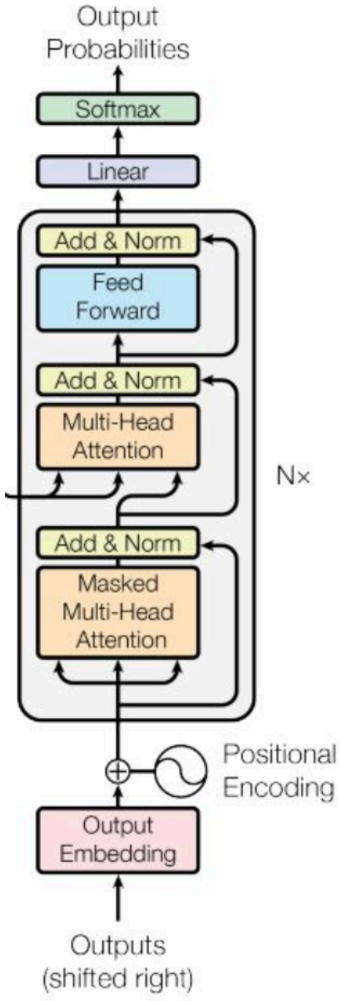


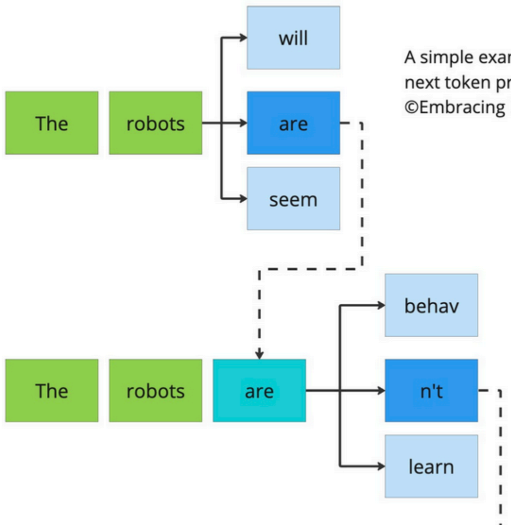


A simple example of autoregressive next token prediction. ©Embracing Enigmas

Large Language Models

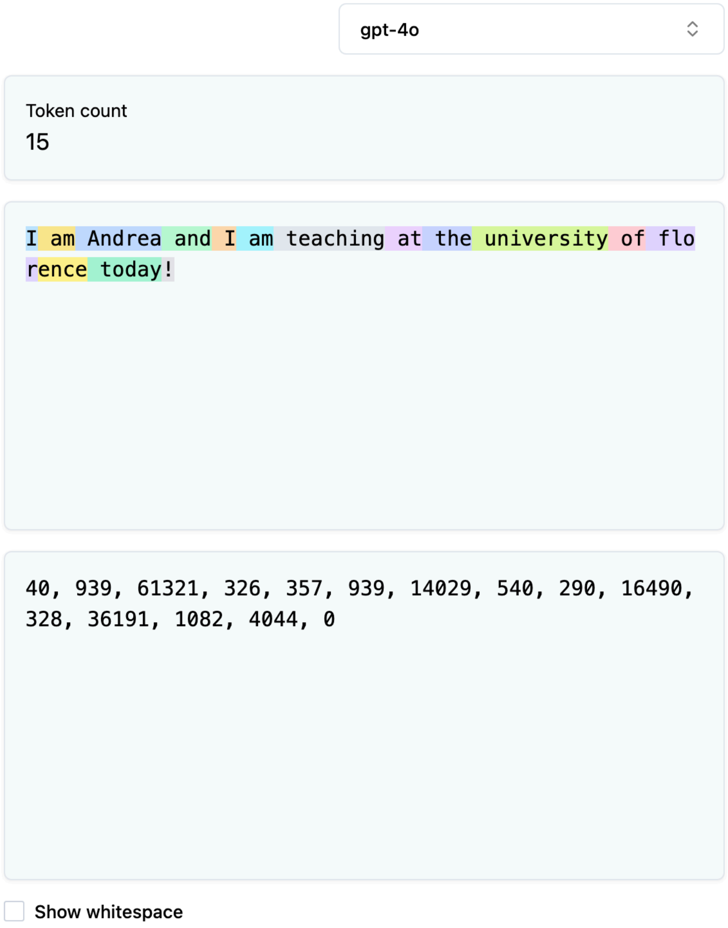


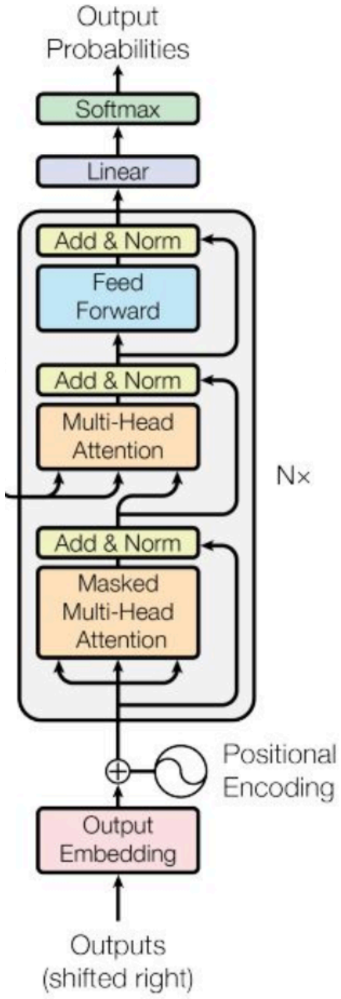


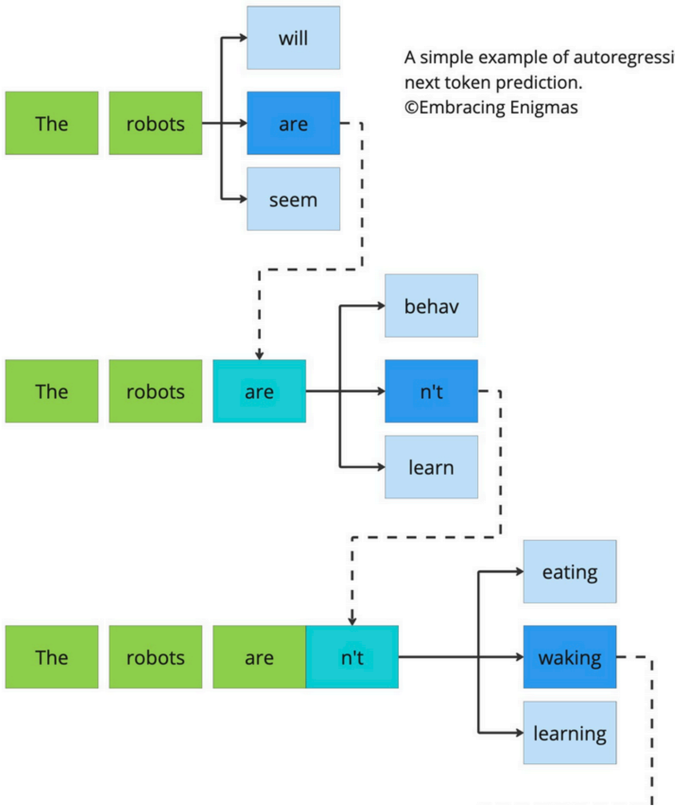


A simple example of autoregressive next token prediction.
©Embracing Enigmas

Large Language Models

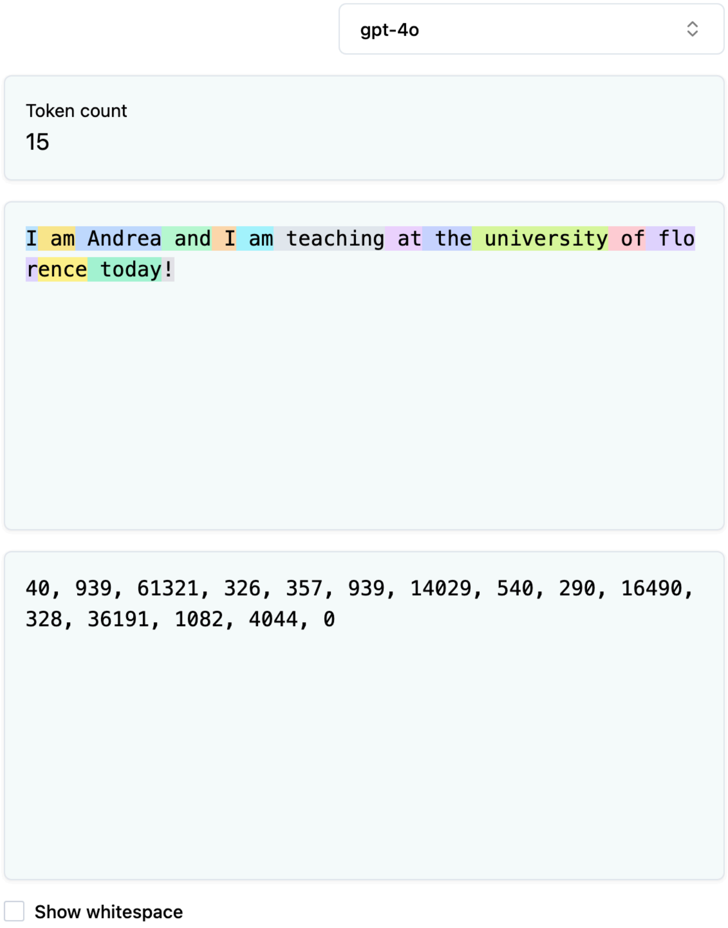


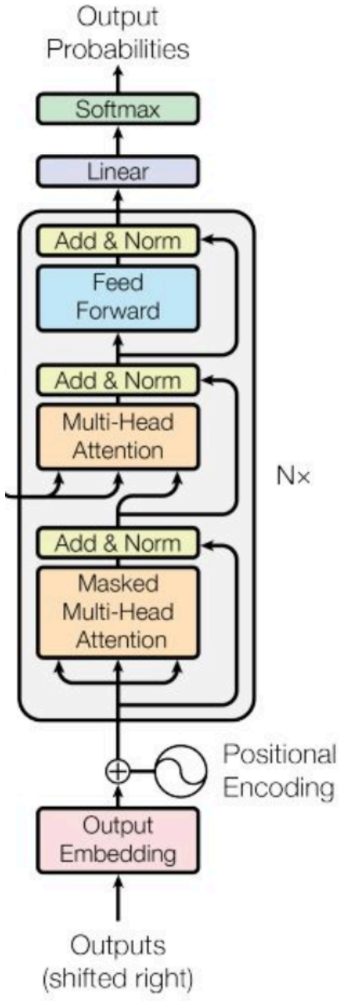


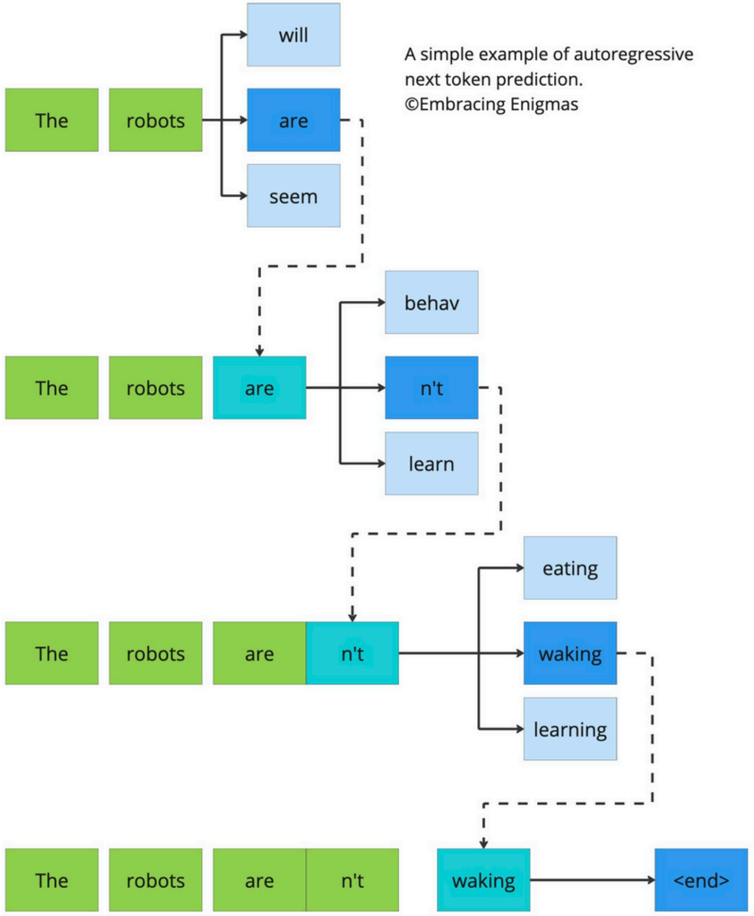


A simple example of autoregressive next token prediction. ©Embracing Enigmas

Large Language Models







The autoregressive objective

An LLM models predict the next token given all previous ones:

$$P(x_t \mid x_1, \ldots, x_{t-1} ; \theta)$$

The Training loss is a Standard cross-entropy, averaged over a huge corpus.

$$L = -\Sigma_t \log P(x_t \mid x{<}t ; \theta)$$

The autoregressive objective

An LLM models predict the next token given all previous ones:

$$\mathbf{P}(\mathbf{x}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-1} \,;\, \theta)$$

The Training loss is a Standard cross-entropy, averaged over a huge corpus.

$$\mathbf{L} = -\,\Sigma_t \log \mathbf{P}(\mathbf{x}_t \mid \mathbf{x}{<}\mathbf{t} \,;\, \theta)$$

LLMs are **stateless**: the "chat" you experience with ChatGPT, Claude etc. is just simulated!

- Every time a new token is generated, <u>the whole conversation is read again</u>!
- The <u>next token is sampled among the whole distribution of the vocabulary</u> and the one with the highest probability is taken (softmax)

From a base model to ChatGPT

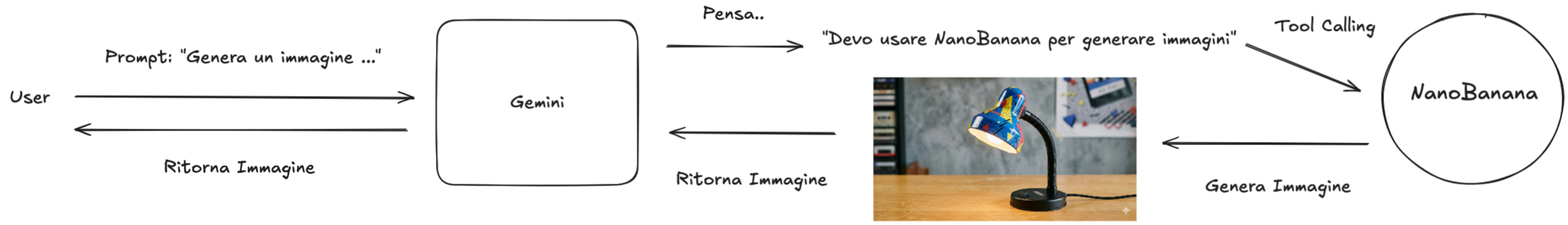
1	**Pre-training**	Next-token prediction on trillions of tokens. Output: a base model that completes text but doesn't follow instructions.
2	**Mid-training**	Continued pre-training on curated, synthetic, or long-context data.
3	**Post-training**	SFT and RLHF
4	**Inference-time scaling**	Let the model think longer before answering. Reasoning models: o1, R1, Gemini Thinking.

LLMs limitations

LLMs are limited in several aspects by design:

- **Missing Updated information**: the knowledge limit come from the data used at the time of retraining
- **Lacking Mathematical exact computation**: They hallucinate 273x1849 or “how many Rs are n raspberry”
- **No Long-context planning**: no memory among calls, one forward pass at a time
- **Limited by language**: no interaction with the “world”

A (partial) solution: Agentic LLMs



PART 02

Agents

What is an agent?

"A system that leverages an AI model to interact with its environment in order to achieve a user-defined objective. It combines reasoning, planning, and the execution of actions — often via external tools — to fulfill tasks."

— HuggingFace Agents Course, Unit 1

Brain

The LLM — reasoning, planning, deciding what to do next.

Body

Tools — APIs, code execution, browsers. The agent's capability space.

System 1 vs System 2

Kahneman, Thinking, Fast and Slow (2011)

System 1

FAST · INTUITIVE · AUTOMATIC

Pattern matching
Immediate output
No verification

≈ a single LLM forward pass.

System 2

SLOW · DELIBERATE · SEQUENTIAL

Reasoning, planning
Tool use
Verification, iteration

≈ an agent: LLM in a loop with tools.

LLMs usage and integration levels

Level	Pattern	Example	Behavior
★☆☆	Router	if llm_decides_A(): a() else: b()	LLM picks a branch
★★☆	Tool caller	run(llm_tool, llm_args)	LLM picks tool + arguments
★★★	Multi-step agent	while llm_should_continue(): step()	LLM controls iteration
★★★	Multi-agent	if llm_trigger(): spawn_agent()	Agents spawn agents

Most production agents live at ★★☆ – ★★★. More agency means more capability and more failure modes.

The ReAct loop

Yao et al., 2022 — arXiv:2210.03629

```
while not done:
    thought     = LLM(context)      # what to do next
    action      = LLM(context)      # tool + arguments
    observation = tool(action)      # actually run it
    context     += (thought, action, observation)

return final_answer
```

Thought

LLM writes its reasoning as text (CoT).

Action

LLM emits a structured tool call.

Observation

Orchestrator runs the tool, returns result into the prompt.

Loop until the LLM emits a final_answer.

Tools — the bridge to the real world

```
def web_search(query: str)        -> list[str]:  ...
def python_repl(code: str)        -> str:        ...
def read_pdf(path: str, page: int) -> str:       ...
def send_email(to: str, body: str) -> bool:      ...
```

Two styles

JSON tool-calling — OpenAI, Anthropic, Gemini APIs.

Code agents — LLM writes Python; sandboxed run (smolagents).

Tool design is the engineering work. Clear names, typed signatures, descriptive docstrings.

A complete agent trace

```
USER:    "Population of the capital of the 2026 World Cup host?"

THOUGHT: I need the 2026 host first.
ACTION:  web_search("2026 FIFA World Cup host country")
OBS:     "USA, Canada, Mexico — joint hosts"

THOUGHT: Three hosts. Get their capitals.
ACTION:  web_search("capital of USA Canada Mexico")
OBS:     "Washington D.C., Ottawa, Mexico City"

THOUGHT: Now populations.
ACTION:  web_search("population Washington DC Ottawa Mexico City 2025")
OBS:     [ ... numbers ... ]

ANSWER:  Co-hosted by USA, Canada, Mexico. Capitals' populations: ...
```

The LLM never knew the answer — but it knew how to find it.

Frameworks to create working Agents

- Smolagents: Minimal, code-first. HuggingFace. Great for practicing and creating small projects
- LangGraph: Graph-based, stateful. Production-oriented.
- LlamaIndex: Strong for RAG and agentic workflows over documents.

Hint: use **Pydantic** and linters (such as ruff) in your projects to make sure **output are structured** in a verifiable python type and all **tool definitions follow standard practices of typing and docstrings**

An working example: https://huggingface.co/spaces/andreagemelli/grandma_secret_sauce

P A R T 0 3

An application example: Agentic Document AI

Documents are not just text

Information in a document is spatial, not just textual.

Layout, tables, figures, charts, checkboxes, signatures — none of this survives a flat OCR pass.

A document in the wild

PDFs, scans, invoices, contracts
Scientific papers with figures and tables
Medical forms, checkboxes, multi-column layouts
Handwritten notes, stamps, signatures

Why "drop a PDF in ChatGPT" fails

1 **Loss of structure**

OCR flattens 2-column layouts; tables become unaligned tokens.

2 **No grounding**

The LLM answers — but you can't verify where in the document it came from.

3 **Visual content ignored**

Questions about charts, figures, checkboxes — the LLM guesses.

Concrete example · DeepSeek-R1 paper

Q: "R1-zero pass@1 at 4000 steps?" ChatGPT-4o → 40% (wrong). Agentic extraction → 60% (correct, region highlighted).

Cite: Khare, Going Beyond OCR+LLM, LandingAI, 2025

A taxonomy of approaches

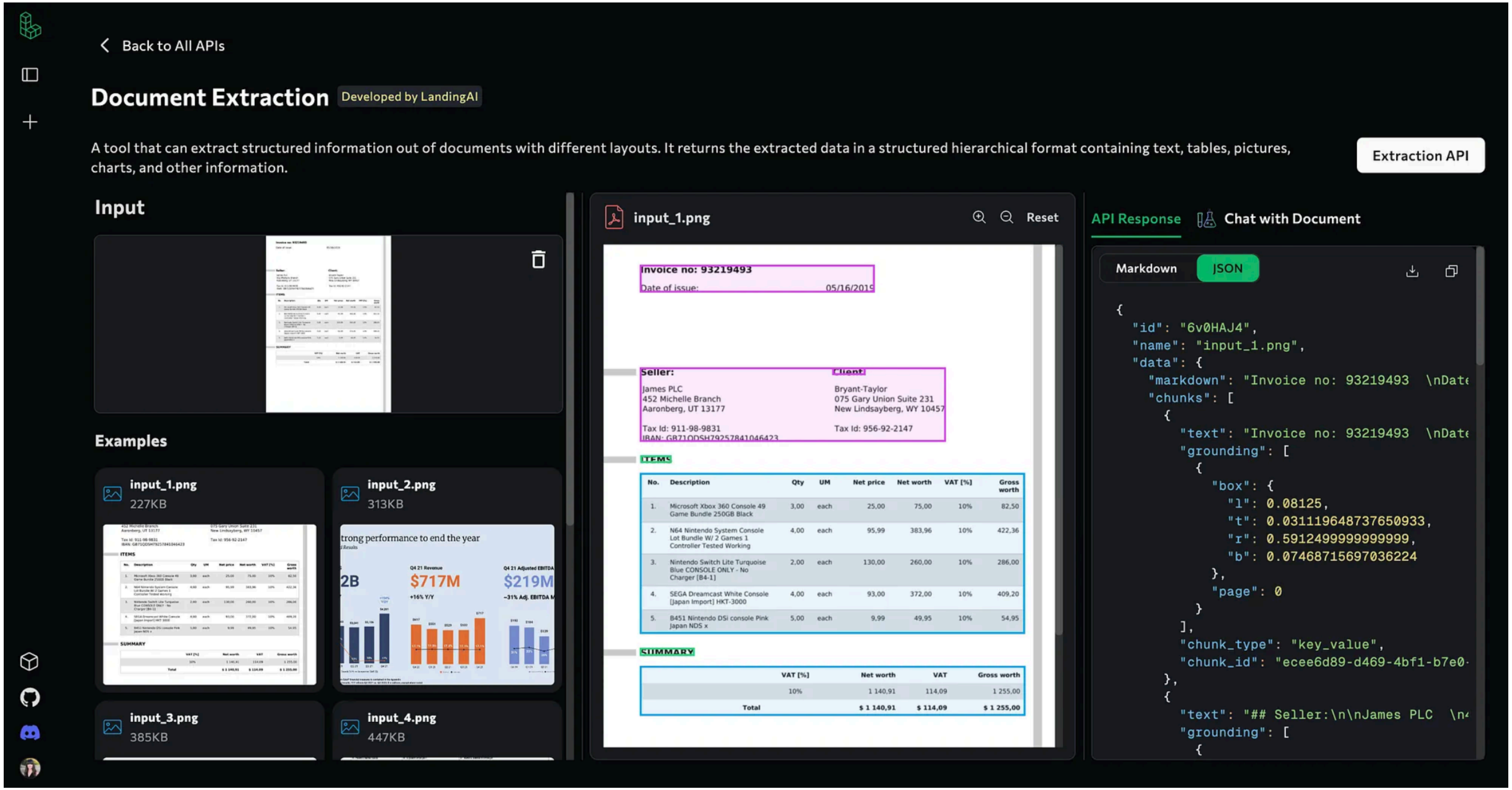
Approach	Examples	Strength	Weakness
Pure OCR	Tesseract, Textract	Cheap, mature	Loses all structure
Layout-aware	LayoutLM, Donut, Nougat	Captures structure	Trained per-domain, brittle
Multimodal LLM	GPT-4V, Claude, Gemini	Flexible, zero-shot	Hallucinates, no grounding
Agentic extraction	LandingAI, Reducto	Verifiable, grounded	Complex, often API-based

Modern systems combine several of these.

Architecture and existing solutions

03 · Document AI

Architecture and existing solutions



Recap

01 LLMs predict the next token — capable, but stateless and toolless.

02 Agents = LLM + loop + tools. They “turn System 1 into System 2.”

03 Document AI in the era of Agents: reasoning + tool use + visual grounding.

References

Foundations

Vaswani et al., Attention Is All You Need (1706.03762)
Karpathy — Let's build GPT (YouTube)
Raschka — State of LLMs 2025

Agents

HuggingFace Agents Course
Yao et al., ReAct (2210.03629)
Kahneman, Thinking, Fast and Slow
smolagents — github.com/huggingface/smolagents
modelcontextprotocol.io

Document AI

Khare, Going Beyond OCR+LLM, LandingAI 2025